Inhaltsverzeichnis

Python Tricks

Python Formatierungen

Oft gebraucht, nie zur Hand, darum hier zum Rauskopieren:

Header:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
```

Function Header:

```
def send_message(sender, recipient, message_body, priority=1) -> int:
   """Send a message to a recipient

   :param str sender: The person sending the message
   :param str recipient: The recipient of the message
   :param str message_body: The body of the message
   :param priority: The priority of the message, can be a number 1-5
   :type priority: integer or None
   :return: the message id
   :rtype: int
   :raises ValueError: if the message_body exceeds 160 characters
   :raises TypeError: if the message_body is not a basestring

   .. todo::
           * Die Routine kann noch keine Territory Codes!
           * Die Routine kann noch keine Aufdröselung von MFC Sammel- codes!

   """
```

der Argparse - Vorspann:

```
import argparse

if __name__=="__main__":
    parser =
argparse.ArgumentParser(formatter_class=argparse.ArgumentDefaultsHelpFormatt
er)
    parser.add_argument(
        "-f",
        "--file",
        help="file to handle",
    )
    parser.add_argument(
        "--identity",
        help="the own identify as email- adress",
```

```
        required=True
    )
    parser.add_argument(
        "--create",
        help="generates key pair or OVERRIDES! existing",
        action='store_true'
    )
    parser.add_argument(
        "--sign",
        help="signs test data and prints signature",
        action='store_true'
    )
    parser.add_argument(
        "--verify",
        help="verify signature by given signature and user"
    )
    parser.add_argument(
        "--user",
        help="verify signature by given signature and user"
    )

    args = parser.parse_args()
```

Python Pip hinter einer self certificate firewall

Einmalig den Truststore unter Abschaltung der SSH- Abfrage installieren

```
pip install --trusted-host pypi.org --trusted-host pypi.python.org --
trusted-host files.pythonhosted.org truststore
```

und dann kann man pip dazu bewegen, den Windows truststore zu benutzen und so das firmen-interne Self Certificate zu akzeptieren

```
pip install tcpc --use-feature=truststore
```

Wenn dann auch noch das Python- Programm (**oder ein submodul während eines pip-Installationsvorgangs**) selber versucht , eine https- Verbindung aufzubauen, fällt das requests-Modul (oder davon abgeleitete Module) ebenfalls auf die Nase:

Erst durch das Nachinstallieren von [1)]

```
pip install pip-system-certs --use-feature=truststore
```

verwendet dann auch Python selber den Windows Truststore

Editable Modules im Compatibily Mode installieren

Gefahr im Verzug: Irgendwie haben sich die Setup-Tools verändert (tbe), was den Effekt hat, das per `pip` lokal installierte Module vom Python- Interpreter nicht mehr gefunden werden.

Nach zwei Stunden der Verzweiflung wurde dann endlich der Parameter- Switch gefunden, der wenigstens noch vorübergehend die Installation ermöglich. Langfristig muss da wohl grundsätzlich an den Modul- Setup- Strukture was Größeres verändert werden. Das riecht nach Stress, aber hier erstmal der Workaround..

```
pip install -e ..\scrlib --config-settings editable_mode=compat
```

Existierendes .venv auf aktuelle Python - Version aktualisieren

Mit folgendem Befehl bringt man .venv auf den [2)] neuesten Stand

```
python3 -m venv --upgrade ENV_DIR
```

Downgrade auf frühere Python- Version (unter Windows)

Manchmal stellt man erschreckend fest, dass manche liebgewonnenen Programme (pyQt6 z.B.) noch nicht unter der neuesten aktuell installierten Version laufen.

Nach viel Gegoogle hilft dann folgendes:

- die ältere Version von python.org herunterladen
- den Installer starten, aber auf `Custom Installation` gehen und beim Setzen der Kreuze tunlichst aufpassen, dass man keine globalen Verweise, Pfade oder ähnliches anrührt, sondern wirklich nur das Softwarepaket irgendwo hin installiert.
- Dann, im Verzeichnis des betroffenen Programmpakets legt man dann ein virtuelles Environment an, aber man benutzt als Python- Interpreter den kompletten Pfad der eben installierten älteren Version

```
C:\Users\skoehler\AppData\Local\Programs\Python\Python39\python.exe -m venv
.venv
```

Damit hat man dann eine virtuelle Umgebung, die auf einer älteren Version läuft als das systemweite Python. In dieser venv- Umgebung kann man dann mit `pip` den ganzen „alten“ Kram wieder installieren 🙂

Venv- Shell aus dem Datei- Explorer starten

Wenn man mal wieder in ein Projekt einsteigen möchte, braucht man dort ja meistens eine Shell in der Venv- Umgebung.

Diese kann man folgendermaßen schnell erreichen:

Man legt sich irgendwo im Suchpfad die Batch- Datei `pyschell.bat` an

pyshell.bat

```
start cmd /k .venv\Scripts\activate.bat
```

Das war's schon. Wenn man jetzt eine Python Shell braucht, geht man im Windows Datei Manager ins gewünschte Verzeichnis und tippt dann oben im URL- Feld, wo man sonst immer nur `cmd` eingetippt hat, statt dessen `pyshell` ein und Schwups, schon öffnet sich ein Kommandozeile in der virtuellen Umgebung.

1)
https://stackoverflow.com/a/57053415
2)
https://stackoverflow.com/a/42405607

From:
http://www.koehlers.de/wiki/ - **Steffen Köhlers Online- Bastelbuch**

Permanent link:
http://www.koehlers.de/wiki/doku.php?id=pc:pythonstype

Last update: **2025/08/20 05:55**