

Python

- Skript - Kontrolle der Sonne
- pip - Ferfügbaren versionen anzeigen lassen
- Python3 Virtual Enviroment

Skript - Kontrolle der Sonne

Um den Sonnen- auf und untergang zu bestimmten, hatte ich zuerst ein Bash Skript benutzt, was ich von einer Seite gefunden hatte.

Nun wurde es in Python ungeschrieben und sieht so aus:

Python:

```
import math
from datetime import datetime, timezone

# Unsere Position
posLaenge = 10.8
posBreite = 53.8

# Notwendige Vorberechnungen
zoneinfo = int(datetime.now().astimezone().utcoffset().total_seconds() / 3600) # Zeitzone in Stunden
T = datetime.now().timetuple().tm_yday # Tag im Jahr
pi = 3.14159265358979323844
rad = pi / 180
h = -(5/6) * rad # Höhe des Sonnenmittelpunkts bei Aufgang: Radius+Refraktion
BreiteRAD = posBreite * rad

# Berechnungen
sonnendekl = 0.409526325277017 * math.sin(0.0169060504029192 * (T - 80.0856919827619))
arccosint = (math.sin(h) - math.sin(BreiteRAD) * math.sin(sonnendekl)) / (math.cos(BreiteRAD) *
math.cos(sonnendekl))
arc2cosint = max(min(arccosint ** 2, 1.0), -1.0)
acoszeit = math.acos(math.sqrt(arc2cosint))

zeitdiff = 12 * acoszeit / pi # KORREKT!

zeitgleich = -0.170869921174742 * math.sin(0.0336997028793971 * T + 0.465419984181394) -
0.129890681040717 * math.sin(0.0178674832556871 * T - 0.167936777524864)
aufgang = 12 - zeitdiff - zeitgleich - (posLaenge / 15) + zoneinfo
untergang = 12 + zeitdiff - zeitgleich - (posLaenge / 15) + zoneinfo

# Ausgabe der Zwischenergebnisse
```

```

print(f"Sonnendeklination: {sonnendekl}")
print(f"Arccosint: {arccosint}")
print(f"Arc2cosint: {arc2cosint}")
print(f"Acoszeit: {acoszeit}")
print(f"Zeitdiff: {zeitdiff}")
print(f"Zeitgleich: {zeitgleich}")
print(f"Aufgang (vor Formatierung): {aufgang}")
print(f"Untergang (vor Formatierung): {untergang}")

# Formatierung
AufgangStunde, AufgangMinute = divmod(int(aufgang * 60), 60)
UntergangStunde, UntergangMinute = divmod(int(untergang * 60), 60)

# Aktualisierte Korrektur
if AufgangStunde == 24:
    AufgangStunde = 0

if UntergangStunde == 24:
    UntergangStunde = 0

# Ausgabe der finalen Ergebnisse
print(f"Aufgang (hh:mm): {AufgangStunde:02d}:{AufgangMinute:02d}")
print(f"Untergang (hh:mm): {UntergangStunde:02d}:{UntergangMinute:02d}")

```

Bash:

```

# Unsere Position
posLaenge="10.8"
posBreite="53.8"

# Notwendige Vorberechnungen
zoneinfo=$(date +%z) # Zeitzone
T=`date +%j` # Tag im Jahr
pi="3.14159265358979323844" # pi=`echo "4*a(1)" | bc -l`
rad=$(echo "${pi}/180" | bc -l)
h=$(echo "-(5/6)*(${rad})" | bc -l) # Höhe des Sonnenmittelpunkts bei Aufgang: Radius+Refraktion
BreiteRAD=$(echo "${posBreite}*${rad}" | bc -l)

# Welcher Tag ist heute?
#echo "Heute ist $(date +%d.%m.%y), der $(date +%j). Tag im Jahr"

```

```

#echo -n "Wir nutzen die Zeitzone $(date +%Z), dies entspricht $(date +%z) und damit "
#echo "${zoneinfo:0:3}"

sonnendekl=`echo "0.409526325277017*s(0.0169060504029192*({T}-80.0856919827619))" | bc -l`
sonnendeklDEG=$(echo "${sonnendekl} / ${rad}" | bc -l)

arccosint=$(echo "(s({h})-s({BreiteRAD})*s({sonnendekl}))/((c({BreiteRAD})*c({sonnendekl})))" | bc -l)
arccosintsign=${arccosint:0:1}
if [ ${arccosintsign} == "-" ]; then
    usesign="+"
else
    usesign="-"
fi
arc2cosint=$(echo "({arccosint}) * ({arccosint})" | bc -l)
acoszeit=$(echo "${pi}/2 ${usesign} a(sqrt({arc2cosint} / (1 - ({arc2cosint}))) ) )" | bc -l)

zeitdiff=$(echo "12*${acoszeit}/${pi}" | bc -l) # KORREKT!

zeitgleich=$(echo "-0.170869921174742*s(0.0336997028793971 * {T} + 0.465419984181394) -
0.129890681040717*s(0.0178674832556871*{T} - 0.167936777524864)" >
aufgang=$(echo "12-({zeitdiff})-({zeitgleich})-({posLaenge}/15){zoneinfo:0:3}" | bc -l)
untergang=$(echo "12+({zeitdiff})-({zeitgleich})-({posLaenge}/15){zoneinfo:0:3}" | bc -l)

if [ ${aufgang:1:1} == "." ]; then
    # Ist ein einstelliges Ergebnis der Form x.xxxx, wir brauchen noch eine 0 vorne
    aufgang=$(echo 0${aufgang})
fi
# Fuer unsere Breitengrade ueberfluessig, nur der Vollstaendigkeit halber:
#if [ ${untergang:1:1} == "." ]; then
# Ist ein einstelliges Ergebnis der Form x.xxxx, wir brauchen noch eine 0 vorne
# untergang=$(echo 0${untergang})
#fi

# Umrechnung in Stunden (trivial) und Minuten (runden!)
AufgangMinute=$(echo "({aufgang} - {aufgang:0:2}) * 60" | bc | xargs printf "%02.0f\n")
if [ ${AufgangMinute} == "60" ]; then
    AufgangMinute="00"
    AufgangStunde=$(echo "${aufgang:0:2} + 1" | bc | xargs printf "%02.0f")
else

```

```
AufgangStunde=${aufgang:0:2}
fi
echo "Aufgang (hh:mm): ${AufgangStunde}:${AufgangMinute}" # Immer ein zweistelliges Ergebnis

UntergangMinute=$(echo "(${untergang} - ${untergang:0:2}) * 60" | bc | xargs printf "%02.0f\n")
if [ ${UntergangMinute} == "60" ]; then
    UntergangMinute="00"
    UntergangStunde=$(echo "${untergang:0:2} + 1" | bc | xargs printf "%02.0f")
else
    UntergangStunde=${untergang:0:2}
fi
echo "Untergang (hh:mm): ${UntergangStunde}:${UntergangMinute}" # Immer ein zweistelliges Ergebnis
```

pip - Ferfügbaren versionen anzeigen lassen

Um alle verfügbaren Pakete von pip anzeigen zu lassen musst du diesen Befehl nutzen:

```
pip3 index versions paho-mqtt
```

Also so in etwa:

```
jj@Sol-Ring ~ % pip3 index versions paho-mqtt
```

```
WARNING: pip index is currently an experimental command. It may be removed/changed in a future release without prior warning.
```

```
paho-mqtt (2.1.0)
```

```
Available versions: 2.1.0, 2.0.0, 1.6.1, 1.6.0, 1.5.1, 1.5.0, 1.4.0, 1.3.1, 1.3.0, 1.2.3, 1.2.2, 1.2.1, 1.2, 1.1, 1.0, 0.9.1, 0.9, 0.4.94, 0.4.92, 0.4.91, 0.4.90
```

```
INSTALLED: 1.6.1
```

```
LATEST: 2.1.0
```

```
jj@Sol-Ring ~ %
```

Python3 Virtual Enviroment

Um das Virtuelle Enviroment zu nutzen, was ab Ubuntu Server 24.04 pflicht wird, musst du folgende Dinge ausführen.

Dabei ist es offensichtlich, das Python3 bennötigt wird und darunter auch das Paket für das Virtuelle Enviroment und da das größte Problem mit pip zusammen hängt (wenn man versucht in Ubuntu 24.04 in pip ein Paket zu installieren ohne ein Virtuelles Enviroment, dann kommt ein Fehler, weil das Virtuelle Enviroment nicht genutzt wird), muss natürlich auch pip installiert sein.

Was ich gerne mache ist, einen extra Ordner für das jeweilige Skript zu nutzen, da das den Sinn und zweck des Virtuellen Enviroments stärkt.

Am Ende muss das Virtuelle Enviroment noch erstellt werden und mit `source .venv/bin/active` genutzt werden.

```
apt install python3 python3-venv python3-pip -y

##### Optional #####
mkdir MQTT_system_status
cd MQTT_system_status/
#####

python3 -m venv .venv
source .venv/bin/activate
```

Um das Virtuelle Enviroment zu verlassen musst du folgenden Befehl eingeben:

```
deactivate
```

