

```
# -*- coding: utf-8 -*-  
"""
```

Created on Thu May 6 21:53:12 2021

```
@author: halvo  
"""
```

```
# -*- coding: utf-8 -*-  
"""
```

Created on Thu May 6 14:47:52 2021

```
@author: Halvor, Kristian  
"""
```

```
#importer biblioteker
```

```
import datetime as dt
```

```
import requests
```

```
import json
```

```
import pandas as pd
```

```
from yr.libyr import Yr
```

```
from astral import LocationInfo
```

```
from astral.sun import sun
```

```
from csv import writer
```

```
# Definerer signal key til "Solar panel production this week"
```

```
panel_key = '8718'
```

```
# Definerer token for å sende info fra CoT
```

```
token = 'eyJhbGciOiJIUzI1NiJ9.eyJqdGkiOiI1NzUxIn0.DeRcDo1IRe0fFV_Iew8WyUbEd02hwzWikjARXvc2oEE'
```

```
def get_currenttime():
```

```
    current_time = dt.datetime.now() # henter dato og tid fra datetime
```

```
    date = current_time.strftime("%d-%m-%y") #Setter dato (format eks. 16-05-21)
```

```
    time = current_time.strftime("%H.%M") # klokkeslett (format eks. 17:30)
```

```
    current_hour = int(current_time.strftime("%H")) # selve time-tallet
```

```
    return [time, current_hour, date]
```

```
def solarenergy_currenthour():
```

```
    # finner klokkeslett for soloppgang, midt på dagen og solnedgang for Trondheim
```

```
    city = LocationInfo("Trondheim", "Norway", "Europe/Oslo", 63.43, 10.39)
```

```
    date_now = dt.date.today() # finner dagens dato fra datetime
```

```
    s = sun(city.observer, date_now, tzinfo=city.timezone)
```

```
    noon = int(s['noon'].strftime("%H"))
```

```
    sunrise = int(s['sunrise'].strftime("%H"))
```

```
    sunset = int(s['sunset'].strftime("%H"))
```

```
    day_length = sunset - sunrise
```

```
    midnight = 0 #definerer midnatt som 00.00
```

```
    current_hour = get_currenttime()[1] # henter klokkeslettet
```

```
    # det er tre timer med full solstråling midt på dagen hvis sola er oppe mer enn eller akkurat 9 timer
```

```
    if (current_hour == noon - 1 or current_hour == noon or current_hour == noon + 1) and day_length >= 9:  
        time_sunfactor = 1
```

```
    # to timer med full stråling når dagen varer i 8 timer
```

```
    elif (current_hour == noon or current_hour == noon + 1) and day_length == 8:  
        time_sunfactor = 1
```

```
    # en time med full solstråling når dagen varer i 7 timer
```

```
    elif current_hour == noon and day_length == 7:  
        time_sunfactor = 1
```

```
    # lite solstråling etter soloppgang og før solnedgang
```

```
    elif (current_hour >= sunrise and current_hour <= sunrise + 1) or (current_hour <= sunset and current_hour >= sunset - 1):  
        time_sunfactor = 0.25
```

```
    # ikke noe sol på natta :O
```

```
    elif (midnight < current_hour < sunrise) or (sunset < current_hour < 23):
```

```
time_sunfactor = 0
```

```
# middels sol ellers på dagen
```

```
else:
```

```
time_sunfactor = 0.6
```

```
return time_sunfactor
```

```
def weather_data():
```

```
# Henter temperaturen og værtypen fra yr for videre bruk for å finne værfaktor og modul effektivitet
```

```
weather_json = Yr(location_name = "Norge/Trøndelag/Trondheim/Trondheim") # henter vær data fra Yr
```

```
now_json = weather_json.now()
```

```
temp_now = int(pd.DataFrame(now_json)["temperature"]["@value"]) # temperatur
```

```
cloud_typ = (pd.DataFrame(now_json)["symbol"]["@name"]) # skydekke
```

```
return [temp_now, cloud_typ]
```

```
def weather_factor():
```

```
cloud_typ = weather_data()[1] #skydekke
```

```
# Definerer lister som bestemmer skydekke (cloud_factor)
```

```
# Dårligere vær fører til lavere cloud_factor som fører til mindre produksjon
```

```
type_1 = ["Clear sky"]
```

```
type_2 = ["Fair"]
```

```
type_3 = ["Partly cloudy", "Light rain showers", "Rain showers", "Heavy rain showers",  
"Light rain showers and thunder", "Rain showers and thunder", "Heavy rain showers and thunder"]  
type_4 = ["Cloudy", "Light rain", "Rain", "Heavy rain", "Light rain and thunder",  
"Rain and thunder", "Heavy rain and thunder"]
```

```
if cloud_typ in type_1:
```

```
cloud_factor = 1
```

```
elif cloud_typ in type_2:
```

```
cloud_factor = 0.8
```

```
elif cloud_typ in type_3:
```

```
cloud_factor = 0.6
```

```
elif cloud_typ in type_4:
```

```
cloud_factor = 0.4
```

```
else:
```

```
cloud_factor = 0.2
```

```
return cloud_factor
```

```
def module_temp_factor():
```

```
temp = weather_data()[0] # Temperature C°
```

```
max_op_temp = 61 # Max Operating Temperature C°
```

```
min_op_temp = -11
```

```
base_mod_eff = 15.7 # Base Module Efficiency %
```

```
temp_coeff = -0.430 # Temperature Coefficient %/C°
```

```
op_temp = 25 # Ideal Operating Temperature C°
```

```
if (temp <= min_op_temp or temp >= max_op_temp):
```

```
mod_eff = 0 # hvis temperaturen er utenfor operating temp-en til panelet produserer panelet ikke noe.
```

```
elif temp > op_temp:
```

```
mod_eff = (base_mod_eff + ((temp-25)*temp_coeff))/100
```

```
# regner ut module efficiency og deler på hundre for å konvertere prosent til desimal
```

```
elif temp < op_temp:
```

```
mod_eff = (base_mod_eff + ((25-temp)*temp_coeff))/100
```

```
else:
```

```
mod_eff = base_mod_eff/100
```

```
# legg til slik at mod eff ikke blir minus
```

```
return mod_eff
```

```
# definerer Funksjonen som blir kalt i hovedprogrammet(planer.py)
```

```
# funksjonen regner ut produksjonen til panelet for den gjellende timen
```

```
def panel_output():
```

```
# Henter in alle relevante faktorer fra de foregående funksjonene
```

```
date = get_currenttime()[2]
```

```
time_sunfactor = solarenergy_currenthour()
```

```
cloud_factor = weather_factor()
```

```
mod_eff = module_temp_factor()
```

```
time = get_currenttime()[0]
```

```
temp = weather_data()[0]
```

```
# Definnerer energien som treffer solcellepanelet og størelsen på panelet
```

```
max_solar_power = 800 # hvor mye solenergi solcellepanelet får på seg W pr m^2
```

```
tot_panel_area = 150 # m^2
```

```
# regner ut kWh output
```

```
W_output = round((max_solar_power * time_sunfactor * cloud_factor * mod_eff * tot_panel_area), 2)
```

```
kWh_output = round((W_output/1000), 2)
```

```
# sender kWh_output til CoT dashboardet
```

```
data_dict = {'Key': panel_key, 'Value': kWh_output, 'Token': token}
```

```
p = requests.put('https://circusofthings.com/WriteValue',
```

```
data = json.dumps(data_dict),
```

```
headers={'Content-Type': 'application/json'})
```

```
# skriver produksjonsverdien til en csv-fil som senere blir brukt
```

```
# til å regne totalproduksjonen for den gjellende dagen.
```

```
with open('temp_csv/Panel_production_hour.csv', 'a') as q:
```

```
prod_df = writer(q)
```

```
prod_df.writerow([date, time, kWh_output])
```

```
q.close()
```

```
with open("csvfiler/to_plot.csv", 'a') as yoo:
```

```
plot_df = writer(yoo, delimiter=";")
```

```
plot_df.writerow([date, time, kWh_output, temp, cloud_factor])
```

```
yoo.close()
```

```
return [kWh_output, W_output]
```