```python
# -*- coding: utf-8 -*-
"""
Created on Mon May  3 12:18:01 2021
@author: krist
"""
# Importerer nødvendige bibliotek
# Denne koden skal kjøres rett før midnatt hver dag
# OBS! må kjøres etter el_forbruk.energy_use()
from csv import writer
import pandas as pd
import requests
import json # importerer biblioteker
import datetime as dt

# Definerer token og signal Key for overføring av verdier til CoT
token = 'eyJhbGciOiJIUzI1NiJ9.eyJqdGkiOiI1NzUxIn0.DeRcDo1IRe0fFV_IEw8WyUbEd02hwzWikjARXvc2oEE'
bill_key = '18287' # info til CoT
panel_key = '8718'
consumption_key = '29260'

nettleie = 0.6 #Kr/kWh

# Definerer funksjon som regner ut total produksjonen til solcelle panelet for den gjellende dagen
def panelprod_day():
    # Finner dato og uke nummer som brukes til å plukke riktige verdier og til lagring av data
    current_date = dt.datetime.now().strftime('%d-%m-%y')
    week_num = dt.datetime.now().isocalendar()[1] # henter dato og ukenummer

    # Henter ut csv-fil som inneholder dagens produksjon slik at verdiene kan hentes ut
    panel_csv = pd.read_csv('temp_csv/Panel_production_hour.csv',index_col='Date').at[current_date, 'kWh'] # leser kWh
outputtet for current_date
    total_production = 0

    # legger sammen alle verdiene for dagen og finner totalproduksjonen for denne dagen
    # Total produksjonen skal brukes til å finne strømregning
    for i in range(len(panel_csv)):
        total_production += panel_csv[i]
    total_production = round(total_production, 2) # kWh

    # Skriver totalproduksjonen til csv-fil med ukenummer og dato for permanent lagring
    with open('csvfiler/Panel_production_day.csv', 'a') as prd:
        prod_df = writer(prd)
        prod_df.writerow([week_num, current_date, total_production])

    return total_production

# Definerer funksjonen som finner differansen (total forbruket) mellom forbruk og produksjon.
# Finner også totale strømregninga for dagen.
def bill():
    # Finner dato og uke nummer som brukes til å plukke riktige verdier og til lagring av data
    # Kunne lagd en egen funksjon for dette for å sleppe gjentatt kode.
    current_date = dt.datetime.now().strftime('%d-%m-%y')
    week_num = dt.datetime.now().isocalendar()[1]

    # Henter ut dagens forbruk i kWh og dagens strømpris i Kr/kWh
    consumption = pd.read_csv('csvfiler/Electricety_consumption.csv', index_col='Date').at[current_date, 'kWh']
    el_price = pd.read_csv('temp_csv/avg_price_day.csv',index_col='Date').at[current_date, 'Kr/kWh'] + nettleie # regner ut
strømpris pris/kWh + nettleie

    # Regner ut total forbruket som er differansen mellom forbruk og produksjon
    total = consumption - panelprod_day()
    # skriver totalt forbruk og dato til csv-fil for permanent lagring
    with open('csvfiler/total_consumption.csv', 'a') as t:
        prod_df = writer(t)
        prod_df.writerow([current_date, total])
```

```
    # Siden overskuddet går til huseier settes total til 0 hvis produksjon er større en forbruk for utregning av strømgegning
    if total <= 0:
        total = 0

    # regner ut strømregninga for gjellende dag
    bill = round(el_price * total, 2)
    # skriver ukenummer, dato og kostnad til csv-fil for permanent lagning
    with open('csvfiler/bill_total.csv', 'a') as b:
        bill_df = writer(b)
        bill_df.writerow([week_num, current_date, bill])
    # Retunerer regnings csv-fil for bruk i neste funksjon
    el_bill_df = pd.read_csv('csvfiler/bill_total.csv', index_col='Week')
    return el_bill_df

# Definerer funksjonen som kalles fra hovedprogrammet
# Denne funksjonen sender strømregningsestimat og strømforbruk for den gjellende uka til CoT
def pay_consm():
    week_num = dt.datetime.now().isocalendar()[1]
    # henter totalforbruket for alle dager den gjellende uka
    consumption_week = pd.read_csv('csvfiler/Electricety_consumption.csv', index_col='Week') # leser ukesforbruk fra csv-fil
    # kaller bill() funksjonen og lagrer den returnerte verdien i en midlertidig csv-fil for å finne regningsestimat for den gjellende
uka
    bill().to_csv('temp_csv/bill_temp.csv')

    # skriver en null verdi til den midlertidige csv-filen slik at man kan bruke sum() funksjonen i neste steg
    # sum funksjonen krever minst to verdier, på mandag ha csv-filen kun en verdi
    with open('temp_csv/bill_temp.csv', 'a') as bt:
        bill_temp_df = writer(bt)
        bill_temp_df.writerow([week_num, '-', 0])

    # Plukker ut verdiene for den gjellende uka
    pay_csv = pd.read_csv('temp_csv/bill_temp.csv', index_col='Week')
    # Regner ut kostand per pers
    pay = round(sum(pay_csv.at[week_num, 'Kr'])/6)

    # sender kostnad per pers til CoT
    data1 = {'Key': bill_key, 'Value': pay, 'Token': token}

    p = requests.put('https://circusofthings.com/WriteValue',
            data = json.dumps(data1),
            headers={'Content-Type': 'application/json'} )

    # Lagrer forbruket i en midlertidig csv-fil for videre manipulasjon uten å endre på denpermanente csv-filen
    consumption_week.to_csv('temp_csv/El_consumption_temp.csv')
    # legger til null verdi av samme grunn som den over
    with open('temp_csv/El_consumption_temp.csv', 'a') as  c:
        consumption_df = writer(c)
        consumption_df.writerow([week_num, '-', 0])

    # Plukker ut verdiene for den gjellende uka
    cons_csv = pd.read_csv('temp_csv/El_consumption_temp.csv', index_col='Week')
    # finner total forbruk for uka
    consumption_this_week = round(sum(cons_csv.at[week_num, 'kWh']), 2) # finner total forbruk for uka

    # Sender forbruket for den gjellende uka til CoT
    data2 = {'Key': consumption_key, 'Value': consumption_this_week, 'Token': token}

    c = requests.put('https://circusofthings.com/WriteValue', # sender totalt ukes forbruk til CoT
            data = json.dumps(data2),
            headers={'Content-Type': 'application/json'} )

    ret = [pay, consumption_this_week]
    print('kost kjørt')
    return ret
```