Heat calculations and graphs

March 27, 2021

[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[3]:
```
data_file = 'data.xlsx'
data_wind = pd.read_excel(data_file)
data_rain = pd.read_excel(data_file, sheet_name=1)
data_radiation = pd.read_excel(data_file, sheet_name=2)
data = data_radiation
```

[4]:
```
min(data_rain['temp'])
```

[4]: 2.2

[5]:
```
min(data_radiation['temp'])
```

[5]: 2.8

[6]:
```
min(data_wind['temp'])
```

[6]: 1.9

[7]:
```
data
```

[7]:

	mm	dd	hh	temp	global radiation
0	1	1	12	12.7	29.98
1	1	1	13	12.9	30.37
2	1	1	14	13.2	27.30
3	1	1	15	13.6	14.98
4	1	1	16	13.9	0.75
…	..	..	..	…	…
8779	12	31	0	13.5	0.00
8780	12	31	0	13.4	0.00
8781	12	31	0	13.7	0.00
8782	12	31	0	13.9	11.38
8783	12	31	0	14.1	30.63

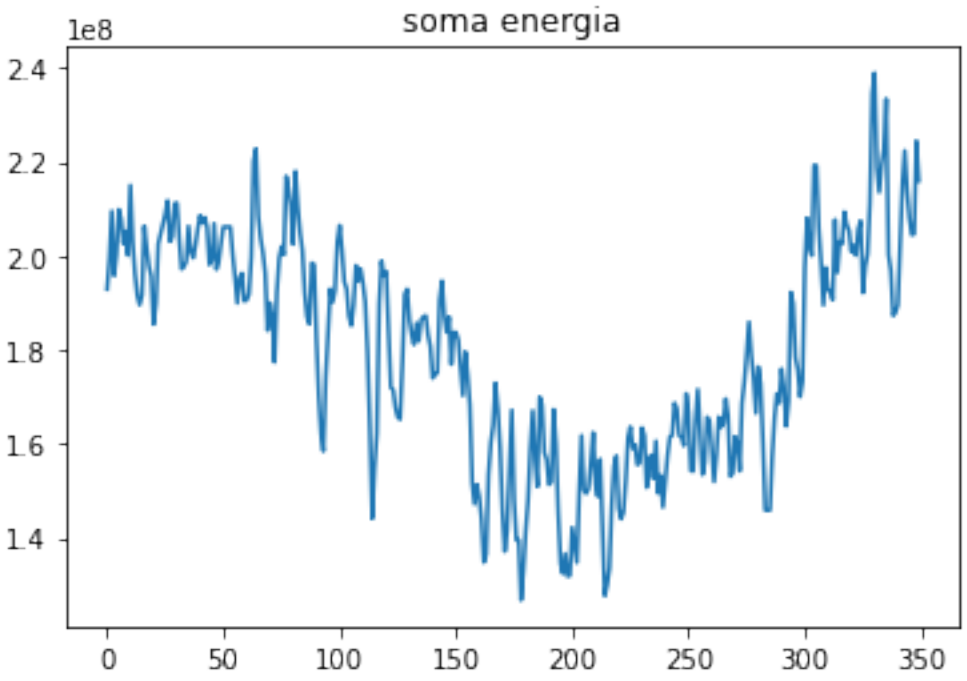
[8784 rows x 5 columns]

[8]:
```
days = data['dd']

hours = data['hh']

temp = data['temp']
```

[9]:
```
#%% water mass

def density (t):

    ro = ((999.8 + 16.9*t -8e-3 * t**2 -46.2e-6 * t**3 + 105.6e-9*t**4 -280.
 ↪5e-12 * t**5) / (1+16.9e-3*t)) / 10**3
    # kg.L

    return (ro)

wat_vol = 60 # L

def mass (den, vol):

    m = den * vol
    # kg

    return (m)

#massess = [mass( density( temp[i] ) , wat_vol ) for i in range( len( temp ) )]
```

[10]:
```
#%% heat


cv = 4018 # J.kg-1.ºC-1

Tf = 45 # ºC


#J considering the boiler efficiency
Q_hour = [mass(density(temp[i]) , wat_vol) * cv * (Tf - temp[i]) * 1.1 for i in␣
 ↪range(len(temp))]


### plot check
# horas = np.arange(len(hours))
# plt.plot(horas, Q_hour)
# plt.show()
# plt.close()
```

[11]:
```
#%% average daily heat necessities

# average heat needed per day
Q_med_day = []


# first day
heat_count = 0
for i in range(12):
    heat_count += Q_hour[i]


ave_heat_day1 = heat_count/12
Q_med_day.append(ave_heat_day1)


somas_q = []
somaq = 0


for k in range(350):

    for i in range(12+24*k , 36 + 24*k):

        somaq += Q_hour[i]

    somas_q.append(somaq)
    somaq = 0

ddd = np.arange(350)
plt.plot(ddd, somas_q)
plt.title('soma energia')
plt.show()
plt.close()

### medias diarias
for val in somas_q :

    media = val/24
    Q_med_day.append(media)

### heat in mega joule
average_heat = []
for val in Q_med_day:
    qMJ = val/10**6
    average_heat.append(qMJ)
```

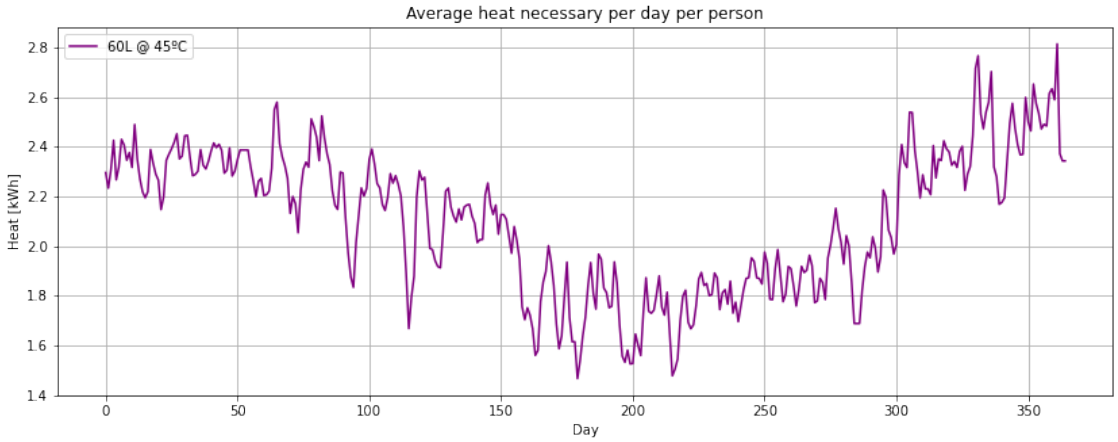


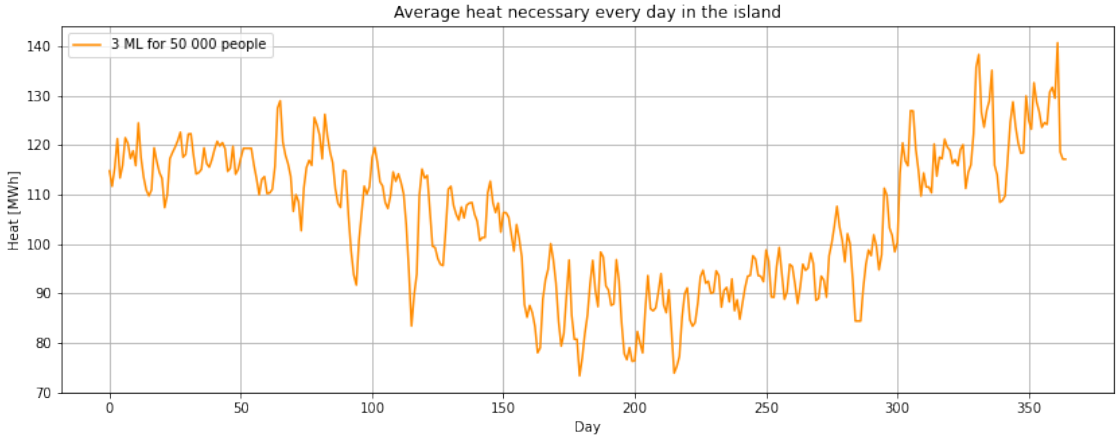
[12]:
```
#%% average daily heat necessities


# average heat needed per day
Q_med_day = []


# first day
heat_count = 0
for i in range(12):
    heat_count += Q_hour[i]


ave_heat_day1 = heat_count/12
Q_med_day.append(ave_heat_day1)
# J


somas_q = []
somaq = 0

dias = 364
```

```
for k in range(dias):

    for i in range(12+24*k , 36 + 24*k):

        somaq += Q_hour[i]

    somas_q.append(somaq)
    somaq = 0


### medias diarias
for val in somas_q :

    media = val/24
    Q_med_day.append(media)

### heat in kWh per person

average_heat = [Q_med_day[i]/3.6e6 for i in range(len(Q_med_day))]
```

[13]:
```
#%% Heat demand by person kWh

d = np.arange(len(average_heat))

plt.figure(figsize=(14,5))
plt.plot(d, average_heat, 'purple', label='60L @ 45ºC')
plt.legend()
plt.title('Average heat necessary per day per person')
plt.xlabel('Day')
plt.ylabel('Heat [kWh]')
plt.grid()
plt.show()
plt.close()

#heat in MWh for 50000 people
island_aver_heat = [average_heat[i]*50 for i in range(len(average_heat))]

plt.figure(figsize=(14,5))
plt.plot(d, island_aver_heat, 'darkorange', label='3 ML for 50 000 people')
```

```
plt.legend()
plt.title('Average heat necessary every day in the island')
plt.xlabel('Day')
plt.ylabel('Heat [MWh]')
plt.grid()
plt.show()
plt.close()
```





[14]:
```
#%% heat necessities total year

anual_island_heat_nec = 0
for i in range(len(island_aver_heat)):

    anual_island_heat_nec += island_aver_heat[i]
    # MWh
```

```
print('Anual heat demand = ' , anual_island_heat_nec, 'MWh')
```

```
Anual heat demand =  38361.06715043907 MWh
```

[19]:
```
#%% --------------- Heat generation -----------
### ---------------- Solar thermic ------------

#new_book = xlrd.open_workbook('excelcarlos.xlsx')

#new_page = new_book.sheet_by_index(3)

new_page = pd.read_excel('excelcarlos.xlsx', sheet_name=3)
```

[21]:
```
new_page
```

[21]:

	Unnamed: 0	Rótulos de Linha	Ep(kWh)	T agua	dt	\
0	dia juliano	d/m	NaN	NaN	NaN	
1	1	1	9222.562076	8.783333	36.216667	
2	2	2	157770.448948	10.687500	34.312500	
3	3	3	130732.858261	9.020833	35.979167	
4	4	4	174906.865603	9.166667	35.833333	
..	…	…	…	…	…	
356	367	26	244904.485839	9.925000	35.075000	
357	368	27	71568.725634	11.525000	33.475000	
358	369	28	29040.543542	NaN	45.000000	
359	370	29	3754.837566	NaN	45.000000	
360	371	31	NaN	NaN	45.000000	

	Q(kJ)	En(kWh)	Ep-En(kWh)	Unnamed: 8	Unnamed: 9	…	\
0	NaN	NaN	NaN	NaN	NaN	…	
1	447420700.0	124283.527778	-115060.965702	NaN	NaN	…	
2	423896625.0	117749.062500	40021.386448	NaN	NaN	…	
3	444486625.0	123468.506944	7264.351317	NaN	NaN	…	
4	442685000.0	122968.055556	51938.810048	NaN	NaN	…	
..	…	…	…	…	…	…	
356	433316550.0	120365.708333	124538.777506	NaN	NaN	…	
357	413550150.0	114875.041667	-43306.316032	NaN	NaN	…	
358	555930000.0	154425.000000	-125384.456458	NaN	NaN	…	
359	555930000.0	154425.000000	-150670.162434	NaN	NaN	…	
360	555930000.0	154425.000000	-154425.000000	NaN	NaN	…	

	Unnamed: 13	Unnamed: 14	Unnamed: 15	m=	60	kg	Unnamed: 19	c	4.118	\
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
..	…	…	…	…	…	…	…	..	…	

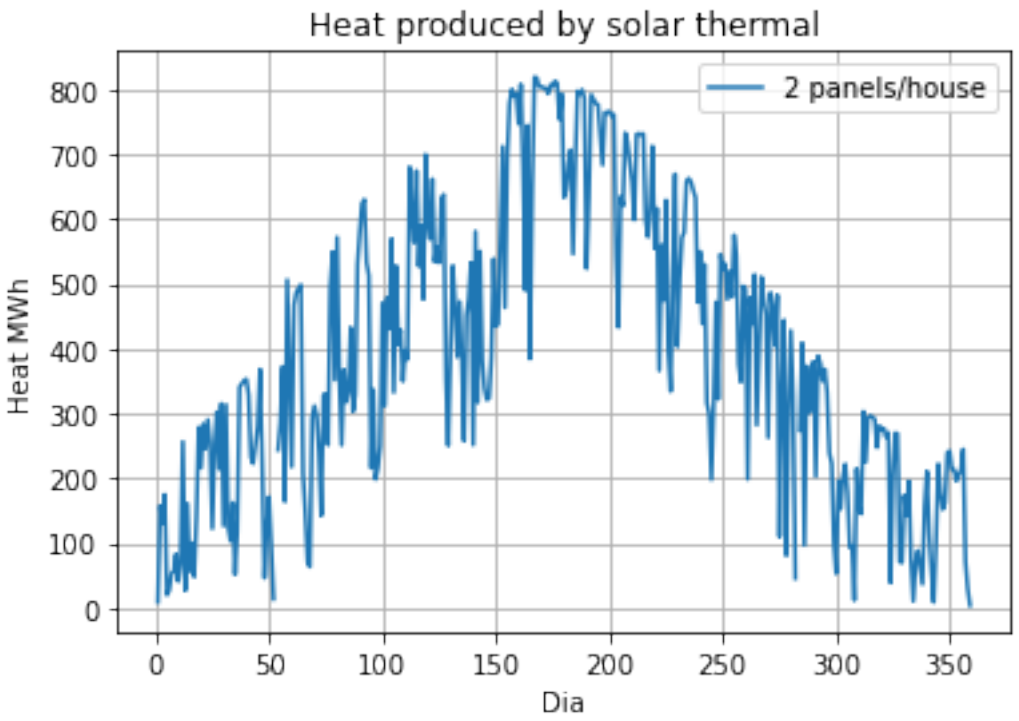
```
356          NaN         NaN         NaN  NaN  NaN  NaN         NaN NaN   NaN
357          NaN         NaN         NaN  NaN  NaN  NaN         NaN NaN   NaN
358          NaN         NaN         NaN  NaN  NaN  NaN         NaN NaN   NaN
359          NaN         NaN         NaN  NaN  NaN  NaN         NaN NaN   NaN
360          NaN         NaN         NaN  NaN  NaN  NaN         NaN NaN   NaN
```

```
     kJ/Kg
0      NaN
1      NaN
2      NaN
3      NaN
4      NaN
..     …
356    NaN
357    NaN
358    NaN
359    NaN
360    NaN

[361 rows x 23 columns]
```

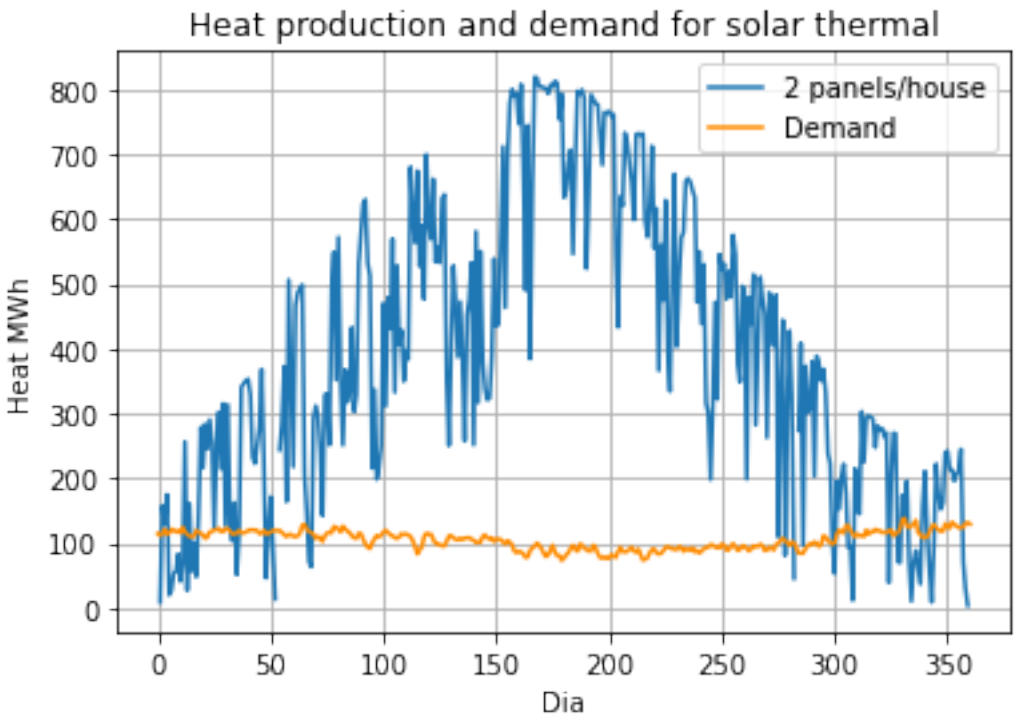
[ ]:

[24]:
```
#%% heat produced by solar thermic by day

# 2 panels
HPST = new_page['Ep(kWh)'] / 10**3
# MWh

island_heat_short = [island_aver_heat[i] for i in range(len(HPST))]
```

[27]:
```
#%%
# plt.figure(figsize=(10,4))
plt.plot(np.arange(len(HPST)), HPST, label = '2 panels/house')
plt.title('Heat produced by solar thermal')
plt.legend()
plt.xlabel('Dia')
plt.ylabel('Heat MWh')
plt.grid()
plt.show()
plt.close()
```



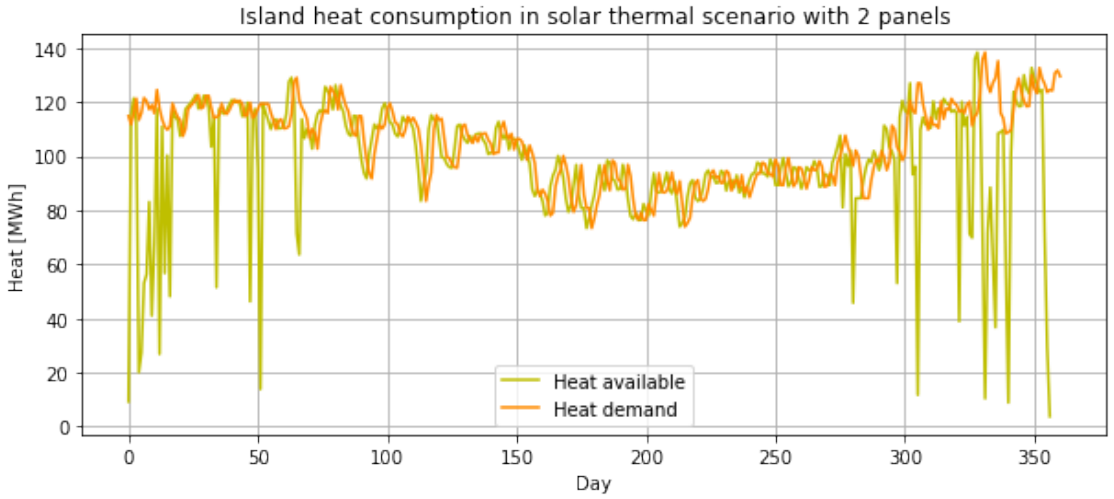
[28]:
```
#%%
# plt.figure(figsize=(10,4))
plt.plot(np.arange(len(HPST)), HPST, label='2 panels/house')
plt.plot(np.arange(len(island_heat_short)), island_heat_short, 'darkorange',␣
 ↪label='Demand')
plt.legend()
plt.title('Heat production and demand for solar thermal')
plt.xlabel('Dia')
plt.ylabel('Heat MWh')
plt.grid()
plt.show()
plt.close()
```

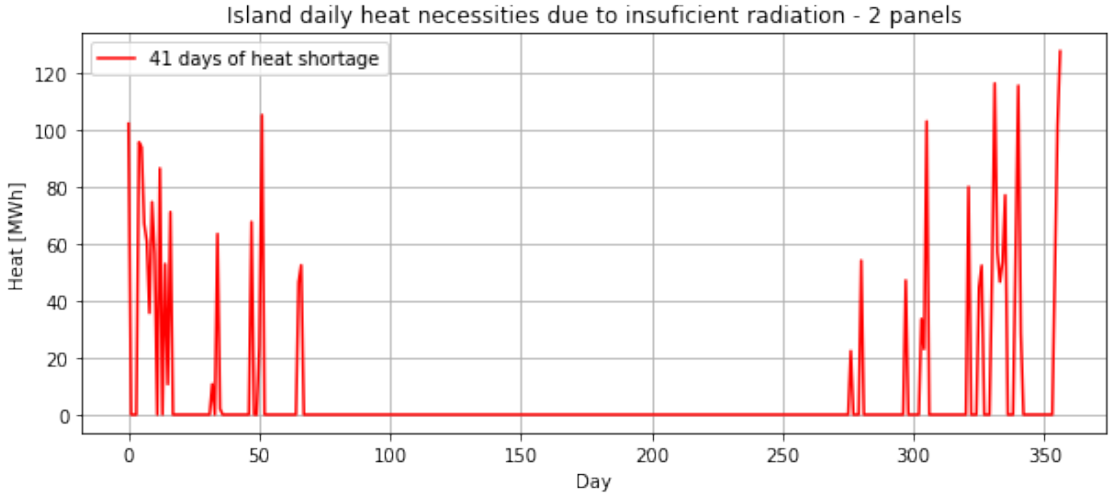


[29]:
```
#%% diference in production and demand

#MWh
dif_calor = [HPST[i] - island_heat_short[i] for i in range(len(HPST))]

cal_cons = []
cal_exc = []
cal_falta = []
cal_insuf = []

days_suf = []
days_insuf = []

for i in range(len(dif_calor)):

    if dif_calor[i] >= 0:

        cal_cons.append(island_heat_short[i])
        cal_exc.append(dif_calor[i])
        cal_insuf.append(0)
        days_suf.append(i)

    if dif_calor[i] < 0:
```

```
        cal_cons.append(HPST[i])
        cal_insuf.append(-dif_calor[i])
        days_insuf.append(i)

ther_prod_tot = 0
for i in range(len(HPST)):

    ther_prod_tot += HPST[i]
    #MWh
print('Heat produced in one year by thermal panels = ' , ther_prod_tot, 'MWh')



# number of days of heat shortage
days_heat_insuf = len(days_insuf)
print ('Days of heat shortage = ' , days_heat_insuf, 'MWh')
# number of days of heat with security of supply
days_heat_suf = len(days_suf)



plt.figure(figsize=(10,4))
plt.plot(np.arange(len(cal_cons)) , cal_cons , 'y', label = 'Heat available')
plt.plot(np.arange(len(island_heat_short)) , island_heat_short, 'darkorange',␣
 ↪label='Heat demand')
plt.legend()
plt.title('Island heat consumption in solar thermal scenario with 2 panels')
plt.xlabel('Day')
plt.ylabel('Heat [MWh]')
plt.grid()
plt.show()
plt.close()


plt.figure(figsize=(10,4))
plt.plot(np.arange(len(cal_insuf)) , cal_insuf , 'r', label =␣
 ↪str(days_heat_insuf) + ' days of heat shortage')
plt.legend()
plt.title('Island daily heat necessities due to insuficient radiation - 2␣
 ↪panels')
plt.xlabel('Day')
plt.ylabel('Heat [MWh]')
plt.grid()
plt.show()
plt.close()
```

```
Heat produced in one year by thermal panels =  nan MWh
Days of heat shortage =  41 MWh
```





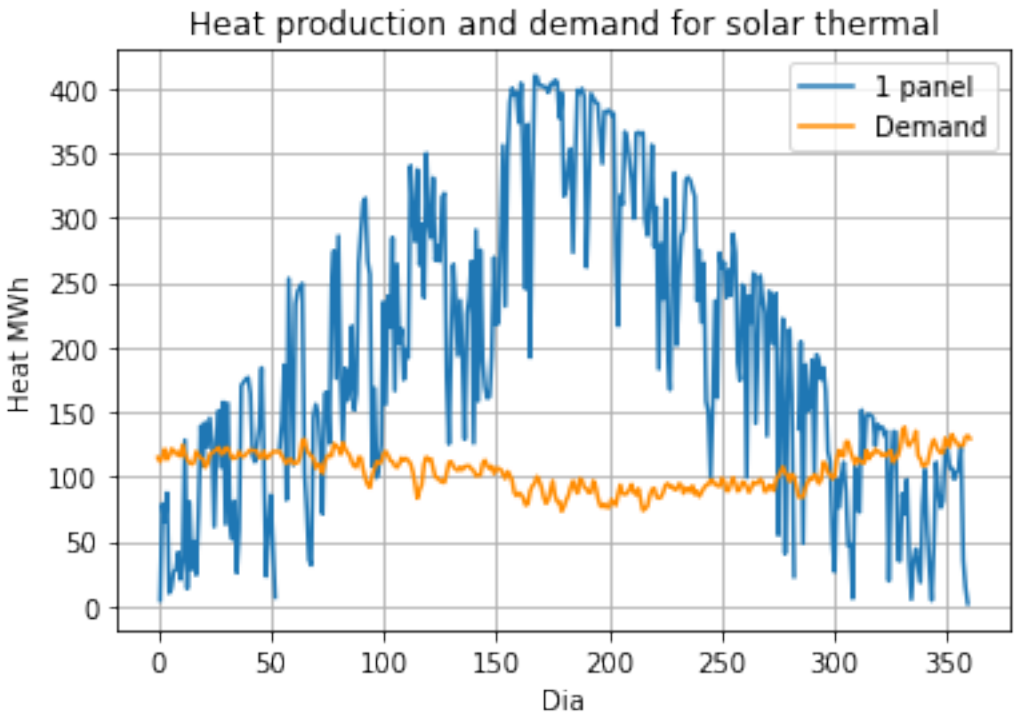
[31]:
```python
#%% 2nd scenario for thermal production
### 1 panel per house

HPST_1p = [HPST[i] / 2 for i in range(len(HPST))]
#MWh

cal_cons_1p = []
cal_exc_1p = []
cal_falta_1p = []
cal_insuf_1p = []
```

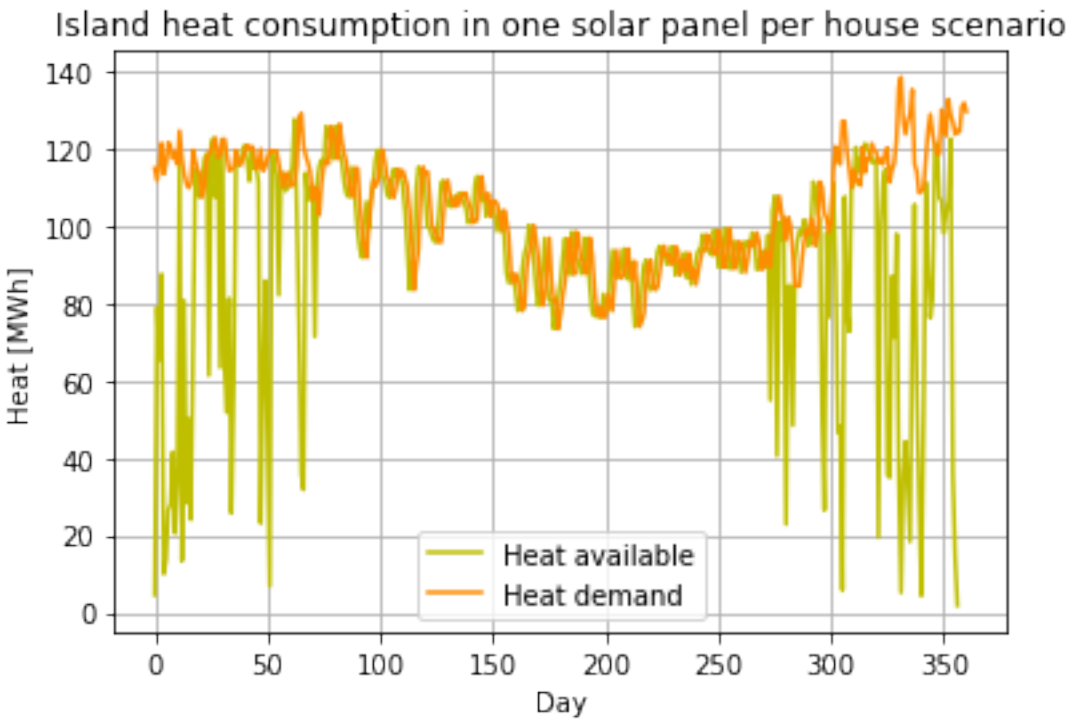
```
days_suf_1p = []
days_insuf_1p = []

### diference in production and demand

#MWh
dif_calor_1p = [HPST_1p[i] - island_heat_short[i] for i in range(len(HPST))]


for i in range(len(dif_calor_1p)):

    if dif_calor_1p[i] >= 0:

        cal_cons_1p.append(island_heat_short[i])
        cal_exc_1p.append(dif_calor_1p[i])
        cal_insuf_1p.append(0)
        days_suf_1p.append(i)

    if dif_calor_1p[i] < 0:

        cal_cons_1p.append(HPST_1p[i])
        cal_insuf_1p.append(-dif_calor_1p[i])
        days_insuf_1p.append(i)

ther_prod_tot_1p = 0
for i in range(len(HPST_1p)):

    ther_prod_tot_1p += HPST_1p[i]
    #MWh
print('Heat produced in one year by thermal panels = ' , ther_prod_tot_1p,␣
 ↪'MWh')



# number of days of heat shortage
days_heat_insuf_1p = len(days_insuf_1p)
print ('Days of heat shortage = ' , days_heat_insuf_1p, 'MWh')
# number of days of heat with security of supply
days_heat_suf_1p = len(days_suf_1p)

#%%
# plt.figure(figsize=(10,4))
plt.plot(np.arange(len(HPST_1p)), HPST_1p, label='1 panel')
plt.plot(np.arange(len(island_heat_short)), island_heat_short, 'darkorange',␣
 ↪label='Demand')
plt.legend()
```

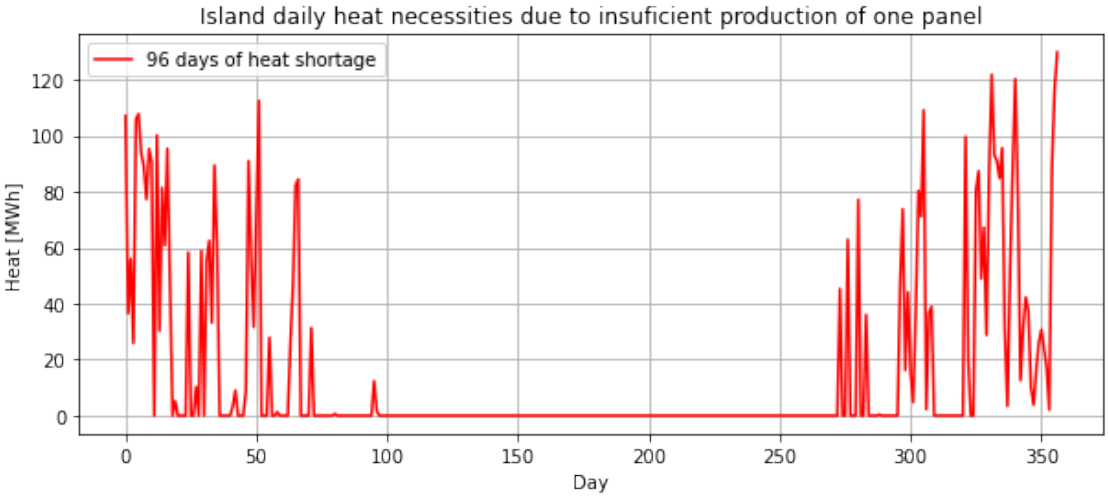
```
plt.title('Heat production and demand for solar thermal')
plt.xlabel('Dia')
plt.ylabel('Heat MWh')
plt.grid()
plt.show()
plt.close()
#%%

# plt.figure(figsize=(10,4))
plt.plot(np.arange(len(cal_cons_1p)) , cal_cons_1p , 'y', label = 'Heat␣
 ↪available')
plt.plot(np.arange(len(island_heat_short)) , island_heat_short, 'darkorange',␣
 ↪label='Heat demand')
plt.legend()
plt.title('Island heat consumption in one solar panel per house scenario')
plt.xlabel('Day')
plt.ylabel('Heat [MWh]')
plt.grid()
plt.show()
plt.close()

#%%

plt.figure(figsize=(10,4))
plt.plot(np.arange(len(cal_insuf_1p)) , cal_insuf_1p , 'r', label =␣
 ↪str(days_heat_insuf_1p) + ' days of heat shortage')
plt.legend()
plt.title('Island daily heat necessities due to insuficient production of one␣
 ↪panel')
plt.xlabel('Day')
plt.ylabel('Heat [MWh]')
plt.grid()
plt.show()
plt.close()


aver_heat_per_short = [average_heat[i] for i in range(len(HPST))]

#%% write data on excel

#results_excel = pd.DataFrame({ 'Heat demand by person [kWh]':␣
 ↪aver_heat_per_short , 'Island heat demand [MWh]': island_heat_short , 'Heat␣
 ↪production [MWh] - One solar panel': HPST_1p , 'Heat consumption [MWh] - One␣
 ↪solar panel': cal_cons_1p , 'Heat still missing - one solar panel [MWh]' :␣
 ↪cal_insuf_1p , 'Heat production [MWh] - Two solar panel': HPST , 'Heat␣
 ↪consumption [MWh] - Two solar panel': cal_cons , 'Heat still missing - two␣
 ↪solar panel [MWh]' : cal_insuf })
```

```
#results_excel.to_excel('ResultsHeatDemandProductionSolar.xlsx', sheet_name =
↪'sheet1', index = False)
```

```
Heat produced in one year by thermal panels =  nan MWh
Days of heat shortage =  96 MWh
```







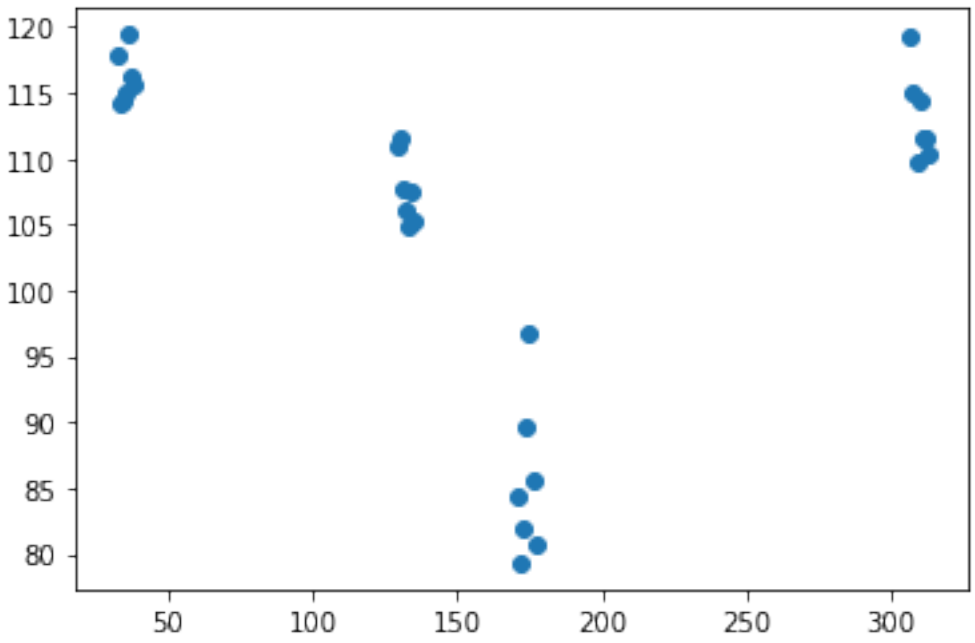
[33]:
```
#%%  Other sources of heat

#heat from waste

# kWh
heat_waste_daily_per = 1.81 * np.ones(366)
```

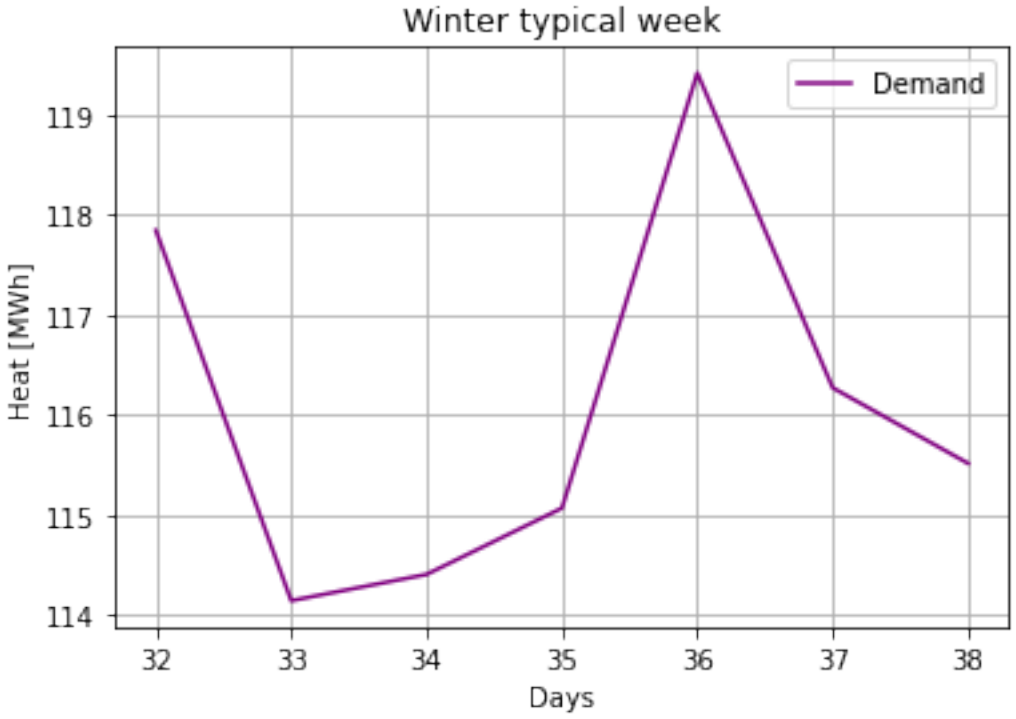
```
# MWh
heat_waste_daily_isl = 1.81 * 50 *np.ones(366)


### heat from biomass

#MWh
heat_biomass_daily_per = 28548 / 366 * np.ones(366)


#%%  typical weeks


###   winter

days_wint_week = np.arange(32,39)

Q_wint_week = [island_heat_short[i] for i in days_wint_week]


### spring

days_spri_week = np.arange(129, 136)

Q_spri_week = [island_heat_short[i] for i in days_spri_week]


### summer

days_summ_week = np.arange(171, 178)

Q_summ_week = [island_heat_short[i] for i in days_summ_week]


### autumn

days_fall_week = np.arange(307, 314)

Q_fall_week = [island_heat_short[i] for i in days_fall_week]


### results comparison

listdays = []
listdays.append(days_wint_week)
listdays.append(days_spri_week)
listdays.append(days_summ_week)
```

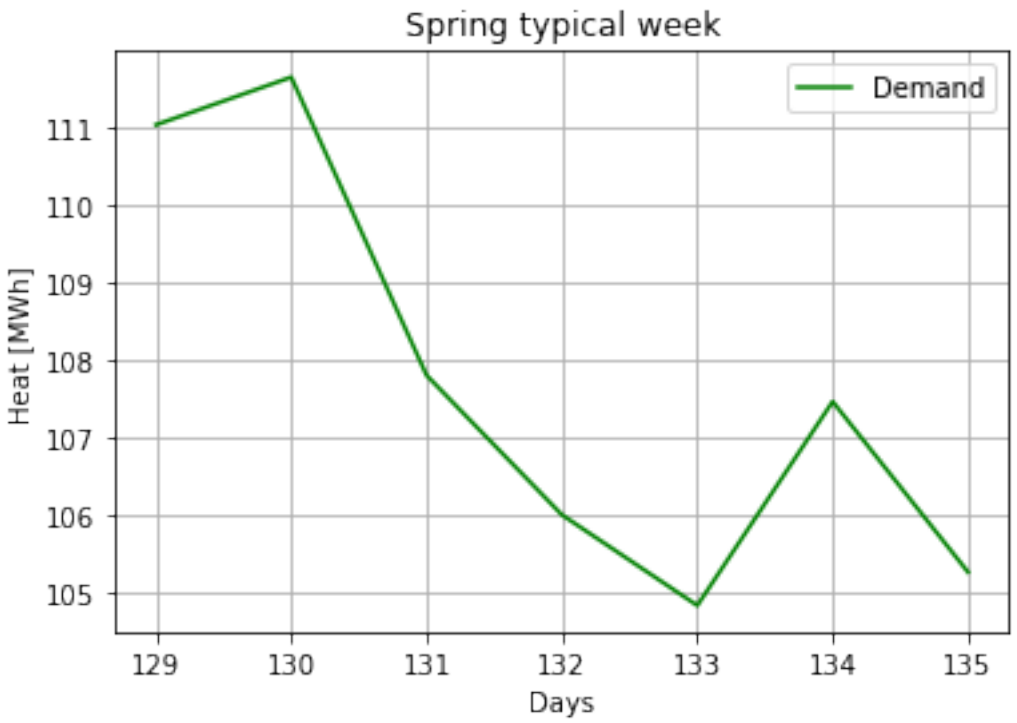
```
listdays.append(days_fall_week)

days_typ_week = []



for i in range(len(listdays)):

    for k in range(len(days_fall_week)):

        days_typ_week.append(listdays[i][k])


#%%
Q_typ_week = [Q_wint_week , Q_spri_week , Q_summ_week , Q_fall_week]


# Q_wint_plot = np.zeros(len(days_typ_week))
# Q_spri_plot = np.zeros(len(days_typ_week))
# Q_summ_plot = np.zeros(len(days_typ_week))
# Q_fall_plot = np.zeros(len(days_typ_week))


# Q_typ_plot = [Q_wint_plot , Q_spri_plot , Q_summ_plot , Q_fall_plot]
#%%

Q_weeks_plot = []

for k in range(len(Q_typ_week)):

    for i in range(len(Q_wint_week)):

        Q_weeks_plot.append(Q_typ_week[k][i])

#%%
### plot of typical weeks


plt.scatter(days_typ_week , Q_weeks_plot)
plt.show()
plt.close()

#%%

plt.plot(days_wint_week , Q_wint_week, 'purple', label = 'Demand')
plt.title('Winter typical week')
plt.xlabel('Days')
```

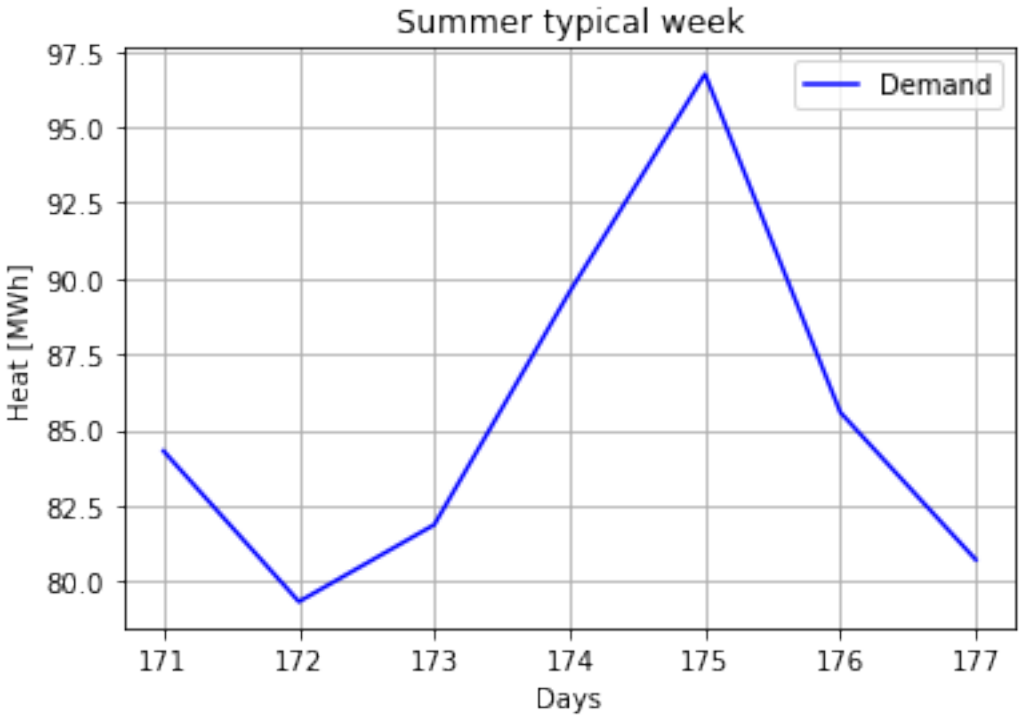
```
plt.ylabel('Heat [MWh]')
plt.legend()
plt.grid()
plt.show()
plt.close()

plt.plot(days_spri_week, Q_spri_week, 'g' , label = 'Demand')
plt.title('Spring typical week')
plt.xlabel('Days')
plt.ylabel('Heat [MWh]')
plt.legend()
plt.grid()
plt.show()
plt.close()

plt.plot(days_summ_week, Q_summ_week, 'b' , label = 'Demand')
plt.title('Summer typical week')
plt.xlabel('Days')
plt.ylabel('Heat [MWh]')
plt.legend()
plt.grid()
plt.show()
plt.close()

plt.plot(days_fall_week, Q_fall_week, 'pink' , label = 'Demand')
plt.title('Autumn typical week')
plt.xlabel('Days')
plt.ylabel('Heat [MWh]')
plt.legend()
plt.grid()
plt.show()
plt.close()


#%% plots of senario: 1 and 2 solar thermic panels


plt.figure(figsize=(12,7))
plt.plot(days_wint_week , Q_wint_week, 'purple', label = 'Demand')
plt.plot(days_wint_week , HPST[32:39], 'r' , label = '2 panels')
plt.plot(days_wint_week , HPST_1p[32:39], 'darkorange' , label = '1 panel')
plt.title('Winter typical week')
plt.xlabel('Days')
plt.ylabel('Heat [MWh]')
plt.legend()
plt.grid()
plt.show()
```

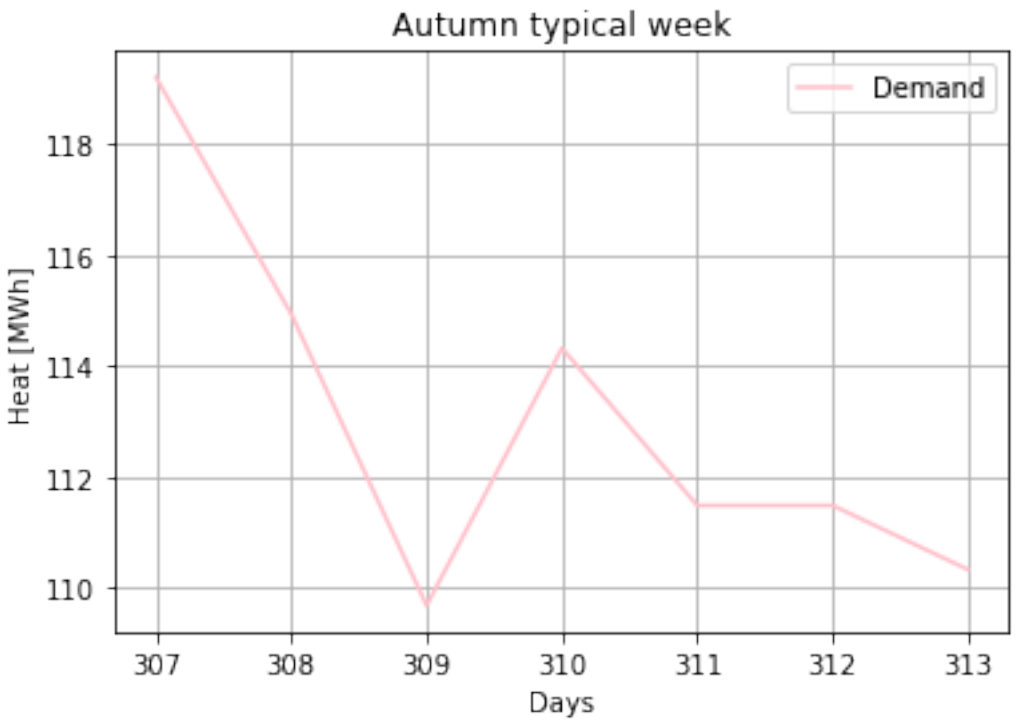
```
plt.close()


plt.figure(figsize=(12,7))
plt.plot(days_spri_week, Q_spri_week, 'g' , label = 'Demand')
plt.plot(days_spri_week , HPST[129:136], 'r' , label = '2 panels')
plt.plot(days_spri_week , HPST_1p[129:136], 'darkorange' , label = '1 panel')
plt.title('Spring typical week')
plt.xlabel('Days')
plt.ylabel('Heat [MWh]')
plt.legend()
plt.grid()
plt.show()
plt.close()

plt.figure(figsize=(12,7))
plt.plot(days_summ_week, Q_summ_week, 'b' , label = 'Demand')
plt.plot(days_summ_week , HPST[171:178], 'r' , label = '2 panels')
plt.plot(days_summ_week , HPST_1p[171:178], 'darkorange' , label = '1 panel')
plt.title('Summer typical week')
plt.xlabel('Days')
plt.ylabel('Heat [MWh]')
plt.legend()
plt.grid()
plt.show()
plt.close()

plt.figure(figsize=(12,7))
plt.plot(days_fall_week, Q_fall_week, 'pink' , label = 'Demand')
plt.plot(days_fall_week , HPST[307:314], 'r' , label = '2 panels')
plt.plot(days_fall_week , HPST_1p[307:314], 'darkorange' , label = '1 panel')
plt.title('Autumn typical week')
plt.xlabel('Days')
plt.ylabel('Heat [MWh]')
plt.legend()
plt.grid()
plt.show()
plt.close()
```

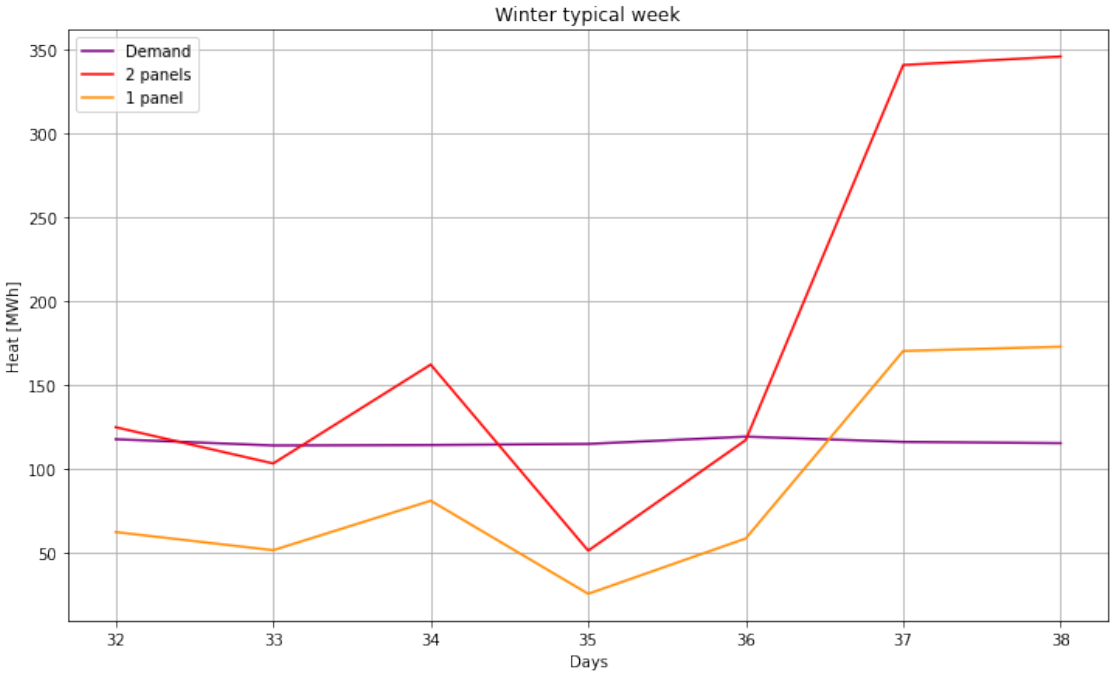

120
115
110
105
100
95
90
85
80
50
100
150
200
250
300

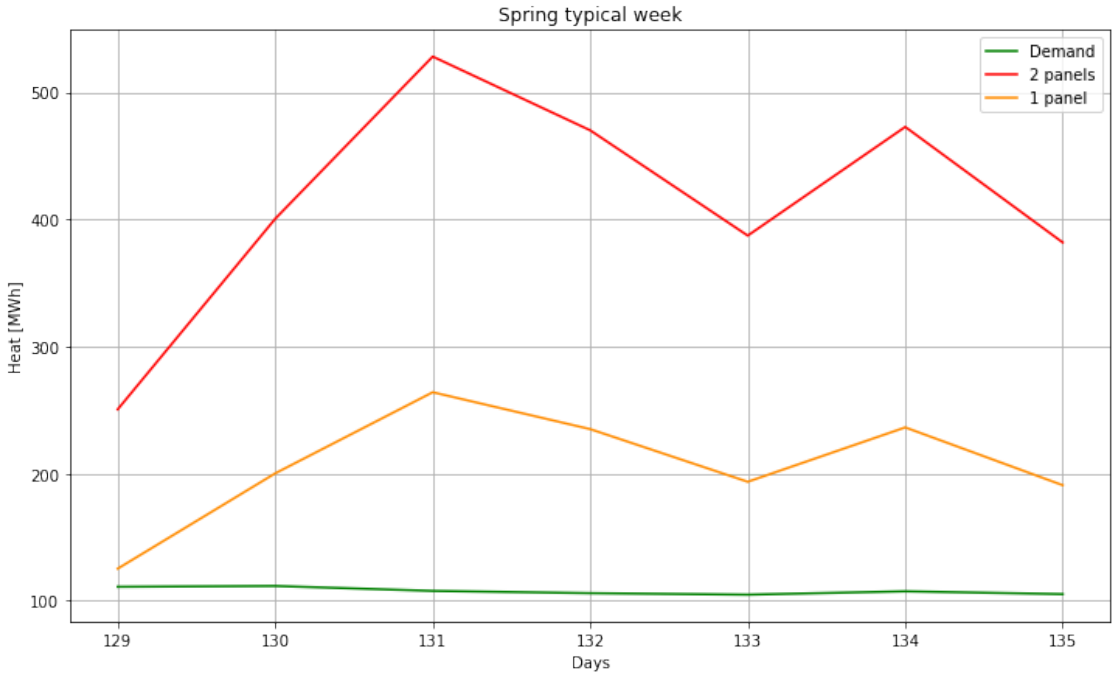

Winter typical week
Demand
Heat [MWh]
119
118
117
116
115
114
32
33
34
35
36
37
38
Days

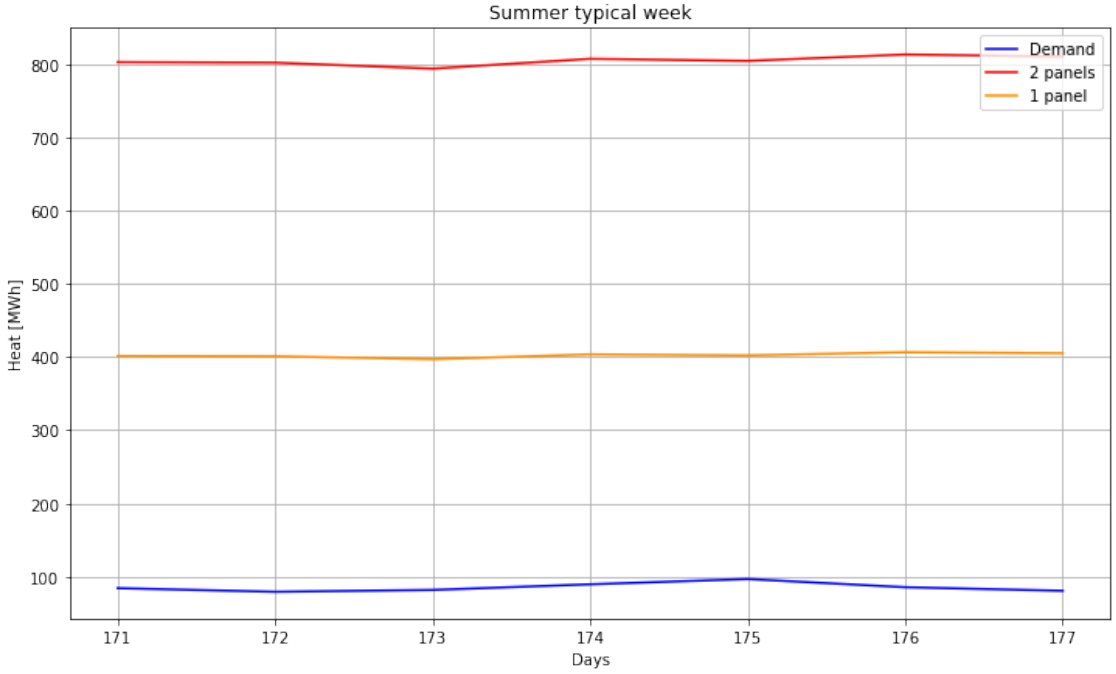

Spring typical week
Demand
Heat [MWh]
111
110
109
108
107
106
105
129
130
131
132
133
134
135
Days

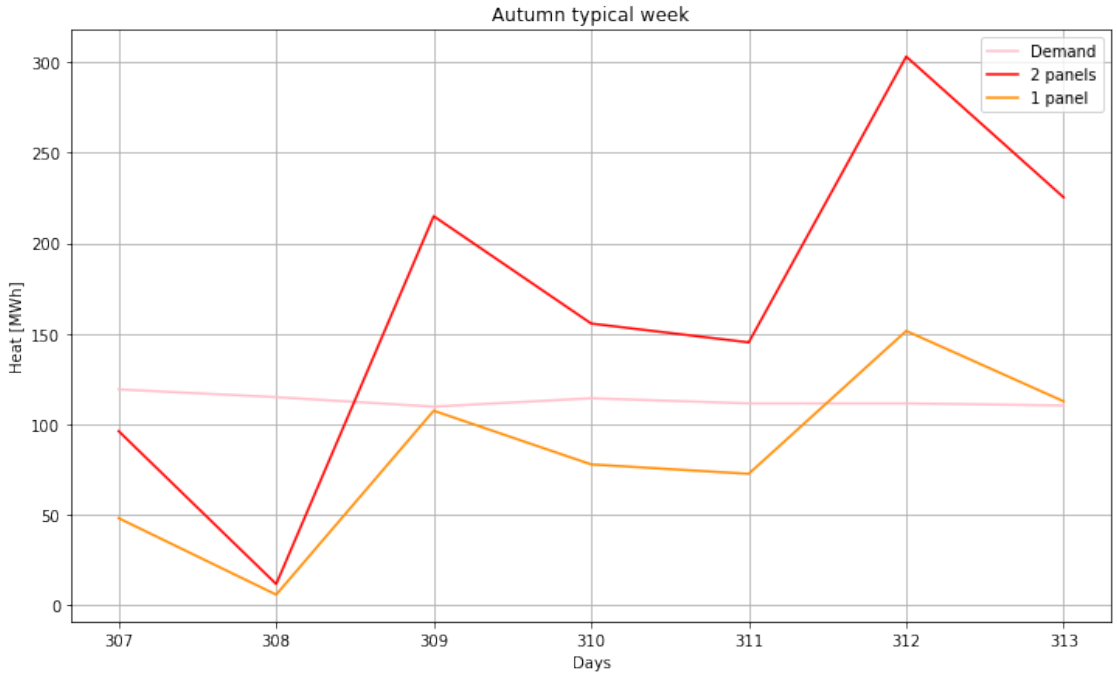

Summer typical week
Demand
Heat [MWh]
97.5
95.0
92.5
90.0
87.5
85.0
82.5
80.0
171
172
173
174
175
176
177
Days


Autumn typical week
Demand
Heat [MWh]
118
116
114
112
110
307
308
309
310
311
312
313
Days


Winter typical week
Demand
2 panels
1 panel
Heat [MWh]
350
300
250
200
150
100
50
32
33
34
35
36
37
38
Days


Spring typical week
Demand
2 panels
1 panel
Heat [MWh]
Days
100
200
300
400
500
129
130
131
132
133
134
135


Summer typical week
Demand
2 panels
1 panel
Heat [MWh]
Days
100
200
300
400
500
600
700
800
171
172
173
174
175
176
177



[ ]:
```
#%% Other energy sources


cenarios = np.loadtxt('energy_combo.txt')

# energy still missing after providing these energy sources

OneSolarPanelWaste = [cenarios[i,0] for i in range(len(cenarios))]
TwoSolarPanelWaste = [cenarios[i,1] for i in range(len(cenarios))]
OneSolarPanelWasteHidro = [cenarios[i,2] for i in range(len(cenarios))]
TwoSolarPanelWasteHidro = [cenarios[i,3] for i in range(len(cenarios))]


# Wind aver production MWh
wind_prod_day_onshore = 12.47 * np.ones(len(cenarios))
wind_prod_day_offshore = 20.31 * np.ones(len(cenarios))

# energy still missing after providing these energy sources

# OneSolarPanelWasteSixOnshore = np.zeros(len(cenarios))

# for i in range(len(cenarios)):

#     if  OneSolarPanelWaste[i] - 6 * wind_prod_day_onshore[i] > 0:
```

```
#         OneSolarPanelWasteSixOnshore = OneSolarPanelWaste[i] - 6 *␣
 ↪wind_prod_day_onshore[i]

#     else:

#         OneSolarPanelWasteSixOnshore[i] = 0

# daycount5 = 0
# for val in OneSolarPanelWasteSixOnshore:

#     if val != 0:

#         daycount5 += 1


#%% experiment of results - adequate nr of wind turbines


### 1 sp + waste + n turbines
aaaa = [OneSolarPanelWaste[i] - 4 * wind_prod_day_onshore[i] for i in␣
 ↪range(len(OneSolarPanelWaste))]
bbbb = [TwoSolarPanelWaste[i] - 4 * wind_prod_day_onshore[i] for i in␣
 ↪range(len(OneSolarPanelWaste))]

ener_falta = []
daycount5 = 0
for val in aaaa:

    if val > 0:

        daycount5 += 1

        ener_falta.append(val)


daycount6 = 0
for val in bbbb:

    if val > 0:

        daycount6 += 1


plt.plot(np.arange(len(aaaa)) , aaaa, label = 'Days of heat shortage = ' +␣
 ↪str(daycount5) + ' days')
plt.title('One panel + waste + 4 onshore')
plt.plot(np.zeros(len(aaaa)) , 'k')
```

```
plt.legend()
plt.show()
plt.close()

plt.plot(np.arange(len(bbbb)) , bbbb, 'g', label = 'Days of heat shortage = ' +␣
 ↪str(daycount6) + ' days')
plt.title('Two panel + waste + 4 onshore')
plt.plot(np.zeros(len(bbbb)) , 'k')
plt.legend()
plt.show()
plt.close()

#%%

##### equals 0
# plt.figure(figsize=(13,6))
# plt.plot(np.arange(len(cenarios)) , OneSolarPanelWasteSixOnshore, 'r' , label␣
 ↪= 'Days of heat shortage = ' + str(daycount5) + ' days')
# plt.title('Heat still missing with one solar panel, waste and 6 wind on shore␣
 ↪turbines')
# plt.xlabel('Days')
# plt.ylabel('Heat MWh')
# plt.legend()
# plt.grid()
# plt.show()
# plt.close()

# OneSolarPanelWasteFourOffshore = [OneSolarPanelWaste[i] - ???␣
 ↪wind_prod_day_offshore[i] for i in range(len(cenarios))]
# TwoSolarPanelWasteSixOnshore = [TwoSolarPanelWaste[i] - ???␣
 ↪wind_prod_day_offshore[i] for i in range(len(cenarios))]
# TwoSolarPanelWasteFourOnshore = [TwoSolarPanelWaste[i] - ???␣
 ↪wind_prod_day_offshore[i] for i in range(len(cenarios))]
# OneSolarPanelWasteHidro??Onshore = [OneSolarPanelWaste[i] - ???␣
 ↪wind_prod_day_offshore[i] for i in range(len(cenarios))]
# OneSolarPanelWasteHidro??Offshore = [OneSolarPanelWaste[i] - ???␣
 ↪wind_prod_day_offshore[i] for i in range(len(cenarios))]
# TwoSolarPanelWasteHidro??Onshore = [OneSolarPanelWaste[i] - ???␣
 ↪wind_prod_day_offshore[i] for i in range(len(cenarios))]
# TwoSolarPanelWasteHidro??Offshore = [OneSolarPanelWaste[i] - ???␣
 ↪wind_prod_day_offshore[i] for i in range(len(cenarios))]



## days of heat shortage
```

```
daycount1 = 0
for val in OneSolarPanelWaste:

    if val != 0:

        daycount1 += 1

daycount2 = 0
for val in TwoSolarPanelWaste:

    if val != 0:

        daycount2 += 1


daycount3 = 0
for val in OneSolarPanelWasteHidro:

    if val != 0:

        daycount3 += 1



daycount4 = 0
for val in TwoSolarPanelWasteHidro:

    if val != 0:

        daycount4 += 1


### plot senarios

#%%
# plt.figure(figsize=(13,6))
plt.plot(np.arange(len(OneSolarPanelWaste)) , OneSolarPanelWaste, 'r' , label =␣
 ↪'Days of heat shortage = ' + str(daycount1) + ' days')
plt.title('Heat still missing with One solar panel and waste')
plt.xlabel('Days')
plt.ylabel('Heat MWh')
plt.legend()
plt.grid()
plt.show()
plt.close()
#%%
```

```
# plt.figure(figsize=(13,6))
plt.plot(np.arange(len(TwoSolarPanelWaste)) , TwoSolarPanelWaste, 'r' , label =␣
 ↪'Days of heat shortage = ' + str(daycount2) + ' days')
plt.title('Heat demand with two solar panel and waste production')
plt.xlabel('Days')
plt.ylabel('Heat MWh')
plt.legend()
plt.grid()
plt.show()
plt.close()

#%%
# plt.figure(figsize=(13,6))
plt.plot(np.arange(len(OneSolarPanelWasteHidro)) , OneSolarPanelWasteHidro, 'r'␣
 ↪, label = 'Days of heat shortage = ' + str(daycount3) + ' days')
plt.title('Heat demand with one solar panel, waste and hydro production')
plt.xlabel('Days')
plt.ylabel('Heat MWh')
plt.legend()
plt.grid()
plt.show()
plt.close()

#%%%
# plt.figure(figsize=(13,6))
plt.plot(np.arange(len(TwoSolarPanelWasteHidro)) , TwoSolarPanelWasteHidro, 'r'␣
 ↪, label = 'Days of heat shortage = ' + str(daycount4) + ' days')
plt.title('Heat demand with two solar panel, waste and hydro production')
plt.xlabel('Days')
plt.ylabel('Heat MWh')
plt.legend()
plt.grid()
plt.show()
plt.close()


#%% heat still missing from thermic solar complemented with eolic turbines

# OneSolarPanel = [cal_insuf_1p[i] for i in range(len(cenarios))]
TwoSolarPanel = [cal_insuf[i] for i in range(len(cenarios))]

nONS = 9

dif_prod_demand_TSP11ONS = [TwoSolarPanel[i] - nONS * wind_prod_day_onshore[i]␣
 ↪for i in range(len(wind_prod_day_onshore))]
```

```
TwoSolarPanel9WONS = dif_prod_demand_TSP11ONS

daycount7 = 0

for i in range(len(TwoSolarPanel9WONS)):

    if dif_prod_demand_TSP11ONS[i] > 0:

        daycount7 += 1

    if dif_prod_demand_TSP11ONS[i] <= 0:

        TwoSolarPanel9WONS[i] = 0

# plt.figure(figsize=(13,6))
plt.plot(np.arange(len(TwoSolarPanel9WONS)) , TwoSolarPanel9WONS, 'r' , label =␣
 ↪'Days of heat shortage = ' + str(daycount7) + ' days')
# plt.plot(np.arange(len(dif_prod_demand_TSP11ONS)) , np.
 ↪zeros(len(dif_prod_demand_TSP11ONS)) , 'k')
plt.title('Heat still missing with Two solar panel and ' + str(nONS) +' ONSHORE␣
 ↪eolic turbine')
plt.xlabel('Days')
plt.ylabel('Heat MWh')
plt.legend()
plt.grid()
plt.show()
plt.close()

#%%
nONS2 = 8

dif_prod_demand_TSP11ONS2 = [TwoSolarPanel[i] - nONS2 *␣
 ↪wind_prod_day_onshore[i] for i in range(len(wind_prod_day_onshore))]

TwoSolarPanel8WONS = dif_prod_demand_TSP11ONS

daycount9 = 0

for i in range(len(TwoSolarPanel8WONS)):

    if dif_prod_demand_TSP11ONS2[i] > 0:

        daycount9 += 1

    if dif_prod_demand_TSP11ONS2[i] <= 0:

        TwoSolarPanel8WONS[i] = 0
```

```
# plt.figure(figsize=(13,6))
plt.plot(np.arange(len(TwoSolarPanel8WONS)) , TwoSolarPanel8WONS, 'r' , label =␣
 ↪'Days of heat shortage = ' + str(daycount9) + ' days')
# plt.plot(np.arange(len(dif_prod_demand_TSP11ONS)) , np.
 ↪zeros(len(dif_prod_demand_TSP11ONS)) , 'k')
plt.title('Heat still missing with Two solar panel and ' + str(nONS2) +'␣
 ↪ONSHORE eolic turbine')
plt.xlabel('Days')
plt.ylabel('Heat MWh')
plt.legend()
plt.grid()
plt.show()
plt.close()

#%%

### off shore
nOFFS = 3

dif_prod_demand_TSP11OFFS = [TwoSolarPanel[i] - nOFFS *␣
 ↪wind_prod_day_offshore[i] for i in range(len(wind_prod_day_offshore))]

TwoSolarPanel7WOFFS = dif_prod_demand_TSP11OFFS

daycount8 = 0
for i in range(len(dif_prod_demand_TSP11OFFS)):

    if val > 0:

        daycount8 += 1

    if val <0 :

        TwoSolarPanel7WOFFS[i] = 0


plt.figure(figsize=(13,6))
plt.plot(np.arange(len(TwoSolarPanel7WOFFS)) , TwoSolarPanel7WOFFS, 'r' , label␣
 ↪= 'Days of heat shortage = ' + str(daycount8) + ' days')
# plt.plot(np.arange(len(dif_prod_demand_TSP11OFFS)) , np.
 ↪zeros(len(dif_prod_demand_TSP11OFFS)) , 'k')
plt.title('Heat still missing with Two solar panel and ' + str(nOFFS) + ' eolic␣
 ↪OFFSHORE turbine')
plt.xlabel('Days')
plt.ylabel('Heat MWh')
```

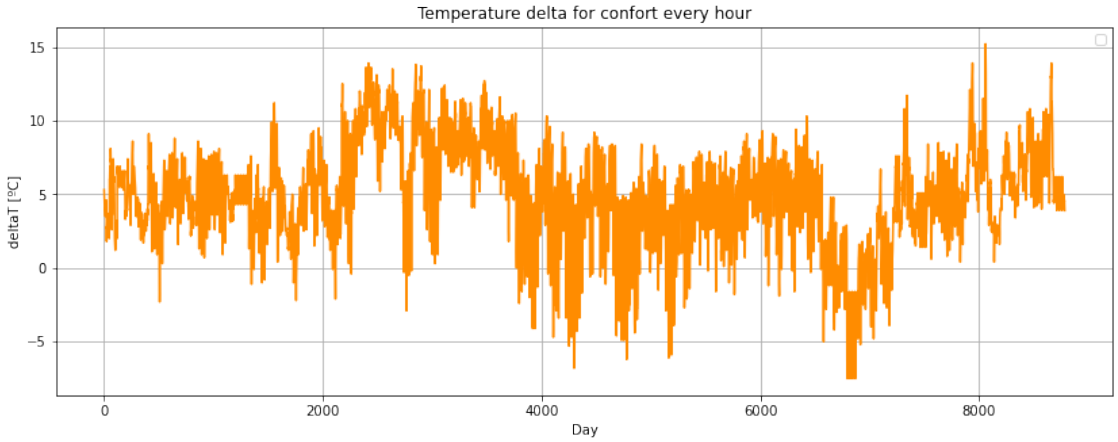
```
plt.legend()
plt.grid()
plt.show()
plt.close()
```

[ ]:

[13]:
```
# Thermal confort
# Calculate the deltaT for each day  to keep the house at the adequate␣
 ↪temperature.
# Summer - Months (4-9) - 25ºC
# Winter - Months (rest) - 18ºC
summer_months = np.arange(4, 10)

delta_T = []

for i in range(len(data)):
    row = data[i:i+1]
    if row['mm'].item() in summer_months:
        delta_t = 25 - row['temp'].item()
    else:
        delta_t = 18 - row['temp'].item()
    delta_T.append(delta_t)

data['delta_T'] = delta_T
```

[14]:
```
plt.figure(figsize=(14,5))
plt.plot(np.arange(len(data['delta_T'])), data['delta_T'], 'darkorange',␣
 ↪label='')
plt.legend()
plt.title('Temperature delta for confort every hour')
plt.xlabel('Day')
plt.ylabel('deltaT [ºC]')
plt.grid()
plt.show()
plt.close()
```

```
No handles with labels found to put in legend.
```



[15]:
```
# Amount of heat required for the air inside the house,
# Q= A deltaT / R
#(R = 2 m2*K*W-1)
R = 2
A = 2*10*10
data['Q'] = (A * data['delta_T']) / R
#data['Q'] = np.absolute(data['Q']) /1000 # /1000 to change to kWh
data['Q'] = data['Q'] /1000 # /1000 to change to kWh
```

[16]:
```
data['Q'][data['Q']<0]=0
```

```
<ipython-input-16-ceaaa0bfd649>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data['Q'][data['Q']<0]=0
```

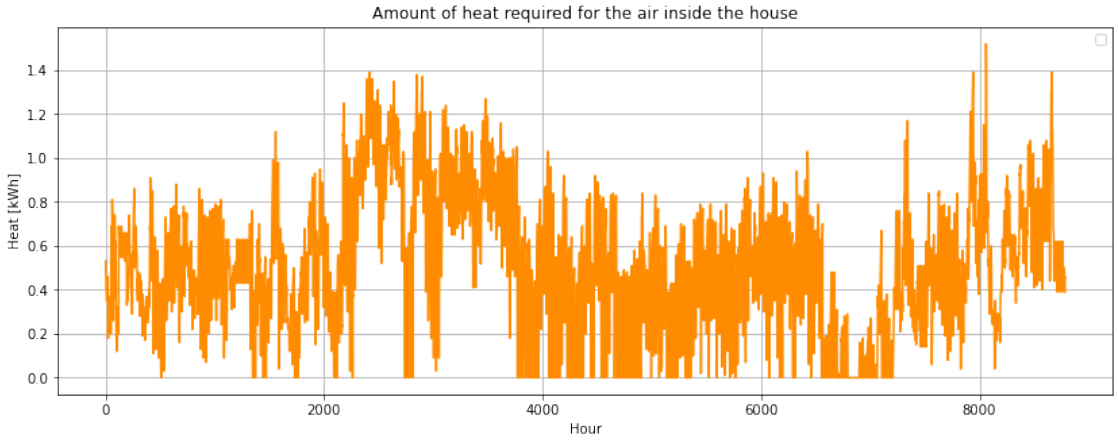
[17]:
```
max(data['Q'])
```

[17]:
```
1.52
```

[18]:
```
plt.figure(figsize=(14,5))
plt.plot(np.arange(len(data['Q'])), data['Q'], 'darkorange', label='')
plt.legend()
plt.title('Amount of heat required for the air inside the house')
plt.xlabel('Hour')
plt.ylabel('Heat [kWh]')
plt.grid()
plt.savefig('Average daily heat in a house hours.png')

plt.show()
```

```
plt.close()
```

```
No handles with labels found to put in legend.
```



[ ]:

[19]:
```
import datetime
datetime.date(month=12, day=1, year=2021)
#data['J_day'] = 1
J_day = np.zeros(len(data))
J_day[0] = 1
for i in range(len(data)):
    if i == len(data)-1:
        break
    row_bef = data[i:i+1]
    row = data[i+1:i+2]
    if row_bef['mm'].item() == row['mm'].item() and row_bef['dd'].item() ==␣
 ↪row['dd'].item():
        J_day[i+1] = J_day[i]
    else:
        J_day[i+1] = J_day[i] + 1
```

[20]:
```
data['J_day'] = J_day
```

[21]:
```
data
```

[21]:

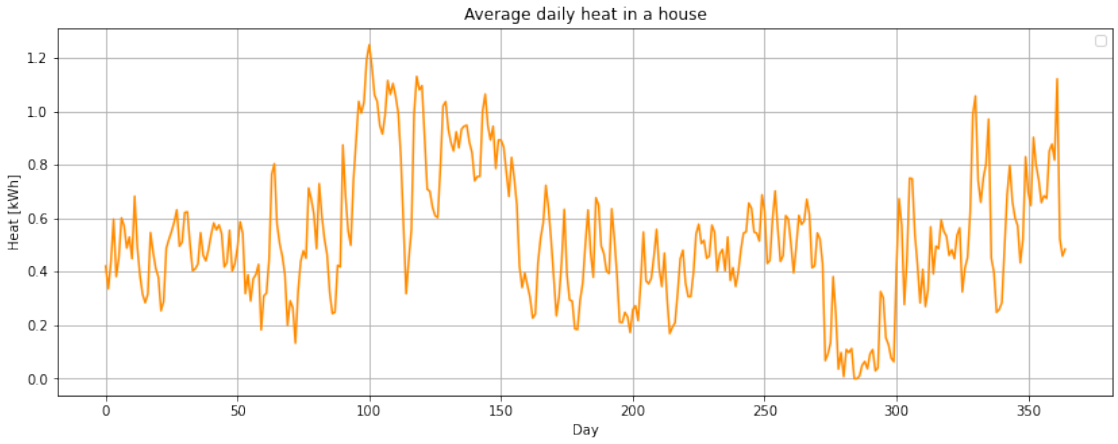
	mm	dd	hh	temp	global radiation	delta_T	Q	J_day
0	1	1	12	12.7	29.98	5.3	0.53	1.0
1	1	1	13	12.9	30.37	5.1	0.51	1.0
2	1	1	14	13.2	27.30	4.8	0.48	1.0
3	1	1	15	13.6	14.98	4.4	0.44	1.0
4	1	1	16	13.9	0.75	4.1	0.41	1.0

```
…    ..   ..   ..   …                …       …    …     …
8779  12   31    0  13.5              0.00      4.5  0.45  365.0
8780  12   31    0  13.4              0.00      4.6  0.46  365.0
8781  12   31    0  13.7              0.00      4.3  0.43  365.0
8782  12   31    0  13.9             11.38      4.1  0.41  365.0
8783  12   31    0  14.1             30.63      3.9  0.39  365.0

[8784 rows x 8 columns]
```

[22]:
```
Q_medi_day = []
for i in np.unique(data['J_day']):
    daily_Q = np.mean(data[data['J_day'] == i]['Q'])
    Q_medi_day.append(daily_Q)
```

[23]:
```
plt.figure(figsize=(14,5))
plt.plot(np.arange(len(Q_medi_day)), Q_medi_day, 'darkorange', label='')
plt.legend()
plt.title('Average daily heat in a house')
plt.xlabel('Day')
plt.ylabel('Heat [kWh]')
plt.grid()
plt.savefig('Average daily heat in a house.png')

plt.show()
plt.close()
```

```
No handles with labels found to put in legend.
```



[24]:
```
np.mean(data['Q'])
```

[24]:
```
0.5268089708561057
```

[25]:
```
sum(data['Q'])
```

[25]: 4627.4900000000325

[26]:
```
data[data['mm']==8][data['dd']==1]
```

```
<ipython-input-26-277d7ab90c88>:1: UserWarning: Boolean Series key will be
reindexed to match DataFrame index.
  data[data['mm']==8][data['dd']==1]
```

[26]:

	mm	dd	hh	temp	global radiation	delta_T	Q	J_day
5100	8	1	0	20.2	0.00	4.8	0.48	213.0
5101	8	1	1	20.0	0.00	5.0	0.50	213.0
5102	8	1	2	19.9	0.00	5.1	0.51	213.0
5103	8	1	3	19.7	0.00	5.3	0.53	213.0
5104	8	1	4	19.5	0.00	5.5	0.55	213.0
5105	8	1	5	19.3	0.00	5.7	0.57	213.0
5106	8	1	6	19.2	0.00	5.8	0.58	213.0
5107	8	1	7	19.1	36.76	5.9	0.59	213.0
5108	8	1	8	19.5	121.30	5.5	0.55	213.0
5109	8	1	9	20.2	250.20	4.8	0.48	213.0
5110	8	1	10	20.6	382.88	4.4	0.44	213.0
5111	8	1	11	21.4	548.11	3.6	0.36	213.0
5112	8	1	12	22.3	854.68	2.7	0.27	213.0
5113	8	1	13	22.7	928.81	2.3	0.23	213.0
5114	8	1	14	22.8	934.45	2.2	0.22	213.0
5115	8	1	15	22.6	838.91	2.4	0.24	213.0
5116	8	1	16	22.2	692.54	2.8	0.28	213.0
5117	8	1	17	21.7	507.83	3.3	0.33	213.0
5118	8	1	18	21.0	300.54	4.0	0.40	213.0
5119	8	1	19	19.9	94.82	5.1	0.51	213.0
5120	8	1	20	18.8	0.00	6.2	0.62	213.0
5121	8	1	21	18.4	0.00	6.6	0.66	213.0
5122	8	1	22	18.2	0.00	6.8	0.68	213.0
5123	8	1	23	18.1	0.00	6.9	0.69	213.0

[27]:
```
winter_week = np.arange(9, 16)
spring_week = np.arange(101, 108)
summer_week = np.arange(207, 214)
```

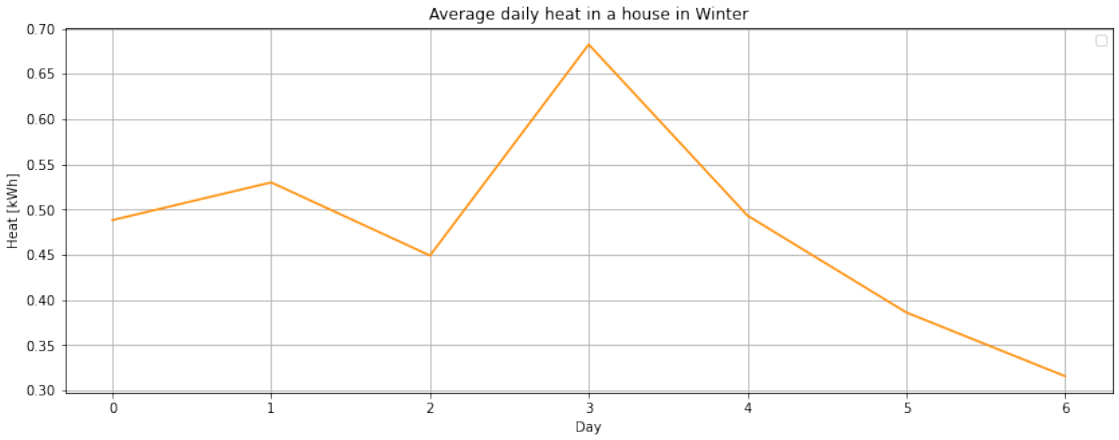
[28]:
```
winter_data = data[data['J_day'].isin(winter_week)]

Q_medi_day_winter = []
for i in np.unique(winter_data['J_day']):
    daily_Q = np.mean(winter_data[winter_data['J_day'] == i]['Q'])
    Q_medi_day_winter.append(daily_Q)
```

[29]:
```
spring_data = data[data['J_day'].isin(spring_week)]

Q_medi_day_spring = []
for i in np.unique(spring_data['J_day']):
    daily_Q = np.mean(spring_data[spring_data['J_day'] == i]['Q'])
    Q_medi_day_spring.append(daily_Q)
```

[30]:
```
summer_data = data[data['J_day'].isin(summer_week)]

Q_medi_day_summer = []
for i in np.unique(summer_data['J_day']):
    daily_Q = np.mean(summer_data[summer_data['J_day'] == i]['Q'])
    Q_medi_day_summer.append(daily_Q)
```

[31]:
```
plt.figure(figsize=(14,5))
plt.plot(np.arange(len(Q_medi_day_winter)), Q_medi_day_winter, 'darkorange',␣
 ↪label='')
plt.legend()
plt.title('Average daily heat in a house in Winter')
plt.xlabel('Day')
plt.ylabel('Heat [kWh]')
plt.grid()
#plt.savefig('Average daily heat in a house(Winter).png')

plt.show()
plt.close()
```

```
No handles with labels found to put in legend.
```

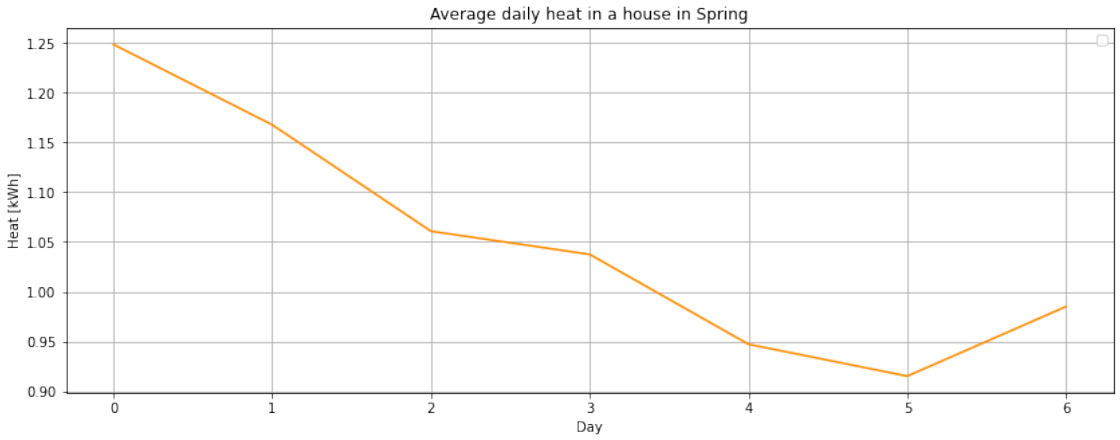


[32]:
```
np.mean(Q_medi_day_winter)
```

[32]:
```
0.47791666666666666
```

[33]:
```
plt.figure(figsize=(14,5))
plt.plot(np.arange(len(Q_medi_day_spring)), Q_medi_day_spring, 'darkorange',␣
 ↪label='')
plt.legend()
plt.title('Average daily heat in a house in Spring')
plt.xlabel('Day')
plt.ylabel('Heat [kWh]')
plt.grid()
#plt.savefig('Average daily heat in a house(Spring).png')
plt.show()
plt.close()
```

```
No handles with labels found to put in legend.
```

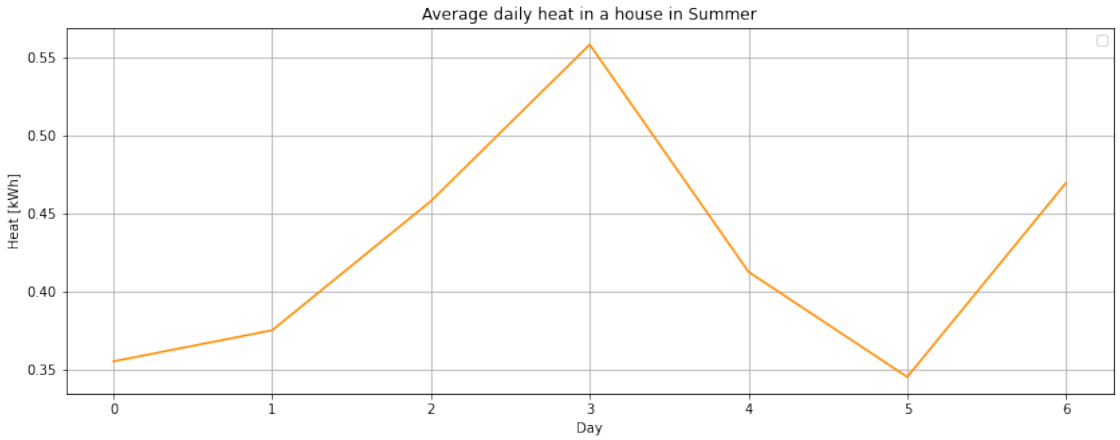


[34]:
```
np.mean(Q_medi_day_spring)
```

[34]:
```
1.0517261904761905
```

[35]:
```
plt.figure(figsize=(14,5))
plt.plot(np.arange(len(Q_medi_day_summer)), Q_medi_day_summer, 'darkorange',␣
 ↪label='')
plt.legend()
plt.title('Average daily heat in a house in Summer')
plt.xlabel('Day')
plt.ylabel('Heat [kWh]')
plt.grid()
#plt.savefig('Average daily heat in a house(Summer).jpg')

plt.show()
plt.close()
```

```
No handles with labels found to put in legend.
```

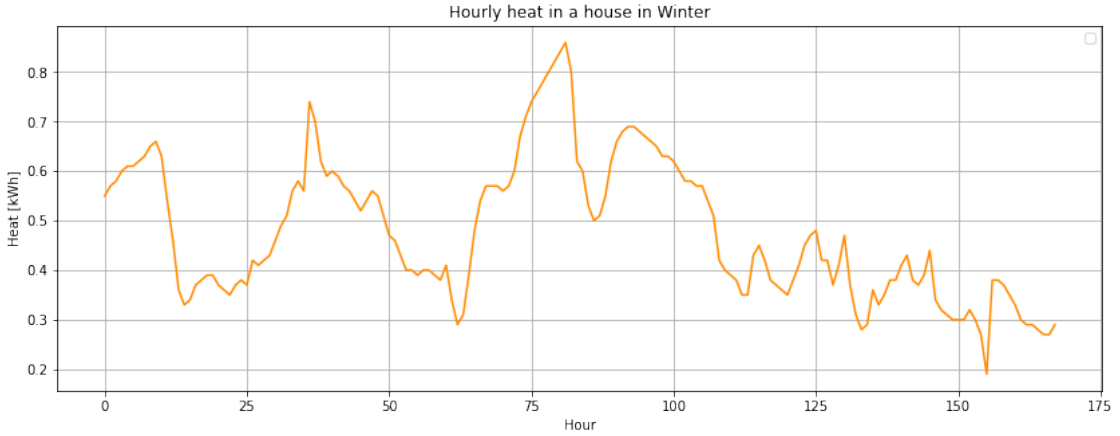


[36]:
```
np.mean(Q_medi_day_summer)
```

[36]:
```
0.4247619047619048
```

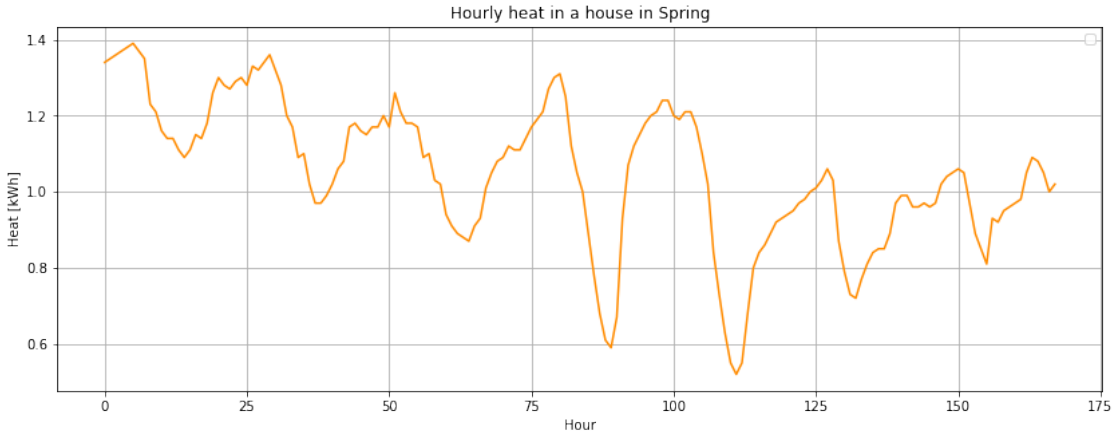
[37]:
```
plt.figure(figsize=(14,5))
plt.plot(np.arange(len(winter_data['Q'])), winter_data['Q'], 'darkorange',␣
↪label='')
plt.legend()
plt.title('Hourly heat in a house in Winter')
plt.xlabel('Hour')
plt.ylabel('Heat [kWh]')
plt.grid()
plt.savefig('Hourly heat in a house(Winter).png')

plt.show()
plt.close()
```

```
No handles with labels found to put in legend.
```



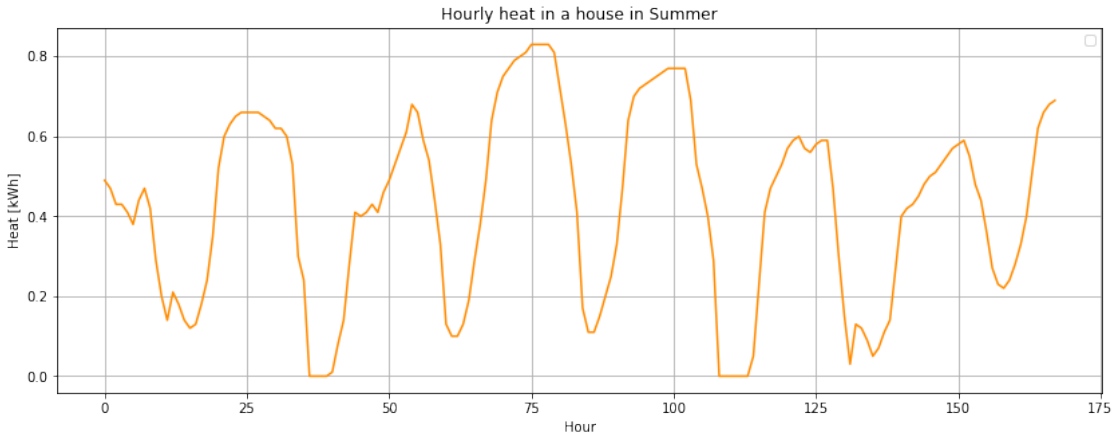
[38]:
```
plt.figure(figsize=(14,5))
plt.plot(np.arange(len(spring_data['Q'])), spring_data['Q'], 'darkorange',␣
 ↪label='')
plt.legend()
plt.title('Hourly heat in a house in Spring')
plt.xlabel('Hour')
plt.ylabel('Heat [kWh]')
plt.grid()
plt.savefig('Hourly heat in a house(Spring).png')

plt.show()
plt.close()
```

```
No handles with labels found to put in legend.
```



[39]:
```
plt.figure(figsize=(14,5))
plt.plot(np.arange(len(summer_data['Q'])), summer_data['Q'], 'darkorange',␣
 ↪label='')
plt.legend()
plt.title('Hourly heat in a house in Summer')
plt.xlabel('Hour')
plt.ylabel('Heat [kWh]')
plt.grid()
plt.savefig('Hourly heat in a house(Summer).png')

plt.show()
plt.close()
```

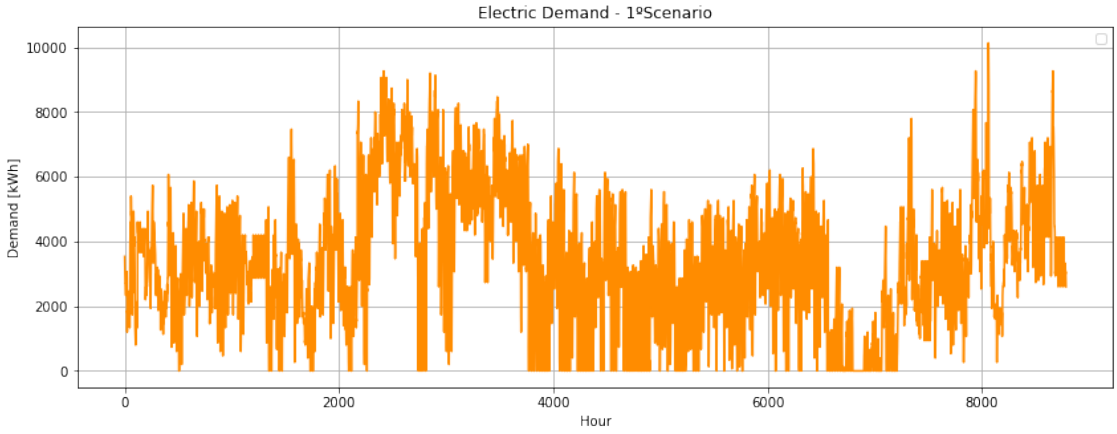
```
No handles with labels found to put in legend.
```



[66]:
```
sum(data['Q'])
data['Q_Island'] = data['Q'] * 20000
```

[67]:
```
plt.figure(figsize=(14,5))
plt.plot(np.arange(len(data['Q'])), data['Q_Island']/3, 'darkorange', label='')
plt.legend()
plt.title('Electric Demand - 1ºScenario')
plt.xlabel('Hour')
plt.ylabel('Demand [kWh]')
plt.grid()
plt.savefig('Electric Demand 1ºScenario.png')

plt.show()
plt.close()
```

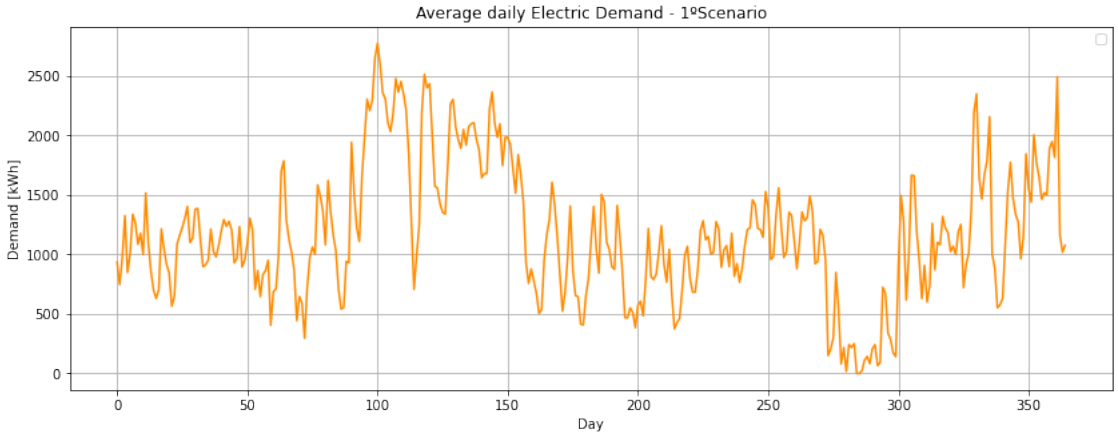
```
No handles with labels found to put in legend.
```



[126]:
```
sce1_data = data
sce1_data['Q_Island'] = sce1_data['Q_Island']/3

Q_medi_day_1st = []
for i in np.unique(data['J_day']):
    daily_Q = np.mean(sce1_data[sce1_data['J_day'] == i]['Q_Island'])
    Q_medi_day_1st.append(daily_Q)
Q_medi_day_1st = np.array(Q_medi_day_1st)
```

[127]:
```
plt.figure(figsize=(14,5))
plt.plot(np.arange(len(Q_medi_day)), Q_medi_day_1st, 'darkorange', label='')
plt.legend()
plt.title('Average daily Electric Demand - 1ºScenario')
plt.xlabel('Day')
plt.ylabel('Demand [kWh]')
plt.grid()
plt.savefig('Average daily Electric Demand 1ºScenario.png')

plt.show()
plt.close()
```

```
No handles with labels found to put in legend.
```



[69]:
```
winter_data = data[data['J_day'].isin(winter_week)]
winter_data['Q_Island'] = winter_data['Q_Island']/3
Q_medi_day_winter = []
for i in np.unique(winter_data['J_day']):
    daily_Q = np.mean(winter_data[winter_data['J_day'] == i]['Q_Island'])
    Q_medi_day_winter.append(daily_Q)


spring_data = data[data['J_day'].isin(spring_week)]
```

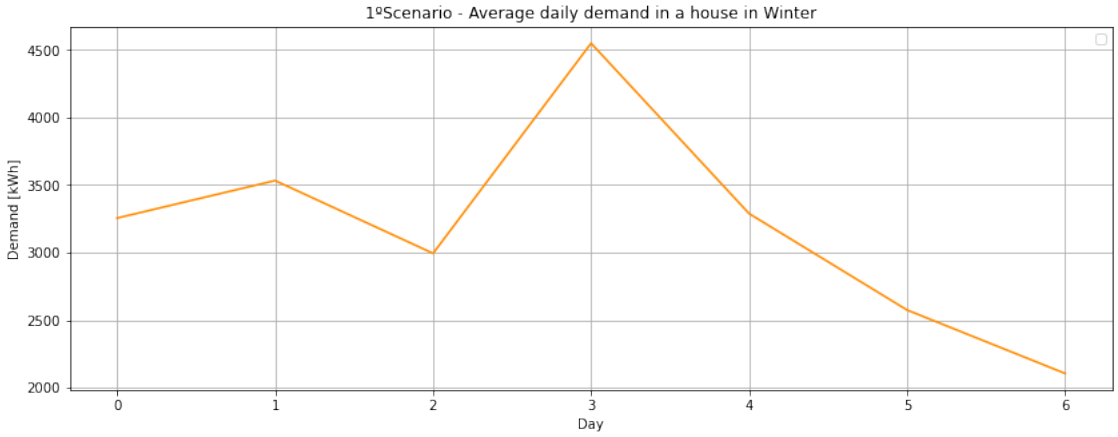
```
spring_data['Q_Island'] = spring_data['Q_Island']/3
Q_medi_day_spring = []
for i in np.unique(spring_data['J_day']):
    daily_Q = np.mean(spring_data[spring_data['J_day'] == i]['Q_Island'])
    Q_medi_day_spring.append(daily_Q)


summer_data = data[data['J_day'].isin(summer_week)]
summer_data['Q_Island'] = summer_data['Q_Island']/3
Q_medi_day_summer = []
for i in np.unique(summer_data['J_day']):
    daily_Q = np.mean(summer_data[summer_data['J_day'] == i]['Q_Island'])
    Q_medi_day_summer.append(daily_Q)
```

```
<ipython-input-69-5569676e9728>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  winter_data['Q_Island'] = winter_data['Q_Island']/3
<ipython-input-69-5569676e9728>:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  spring_data['Q_Island'] = spring_data['Q_Island']/3
<ipython-input-69-5569676e9728>:18: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  summer_data['Q_Island'] = summer_data['Q_Island']/3
```

[70]:
```
plt.figure(figsize=(14,5))
plt.plot(np.arange(len(Q_medi_day_winter)), Q_medi_day_winter, 'darkorange',␣
 ↪label='')
plt.legend()
plt.title('1ºScenario - Average daily demand in a house in Winter')
plt.xlabel('Day')
plt.ylabel('Demand [kWh]')
plt.grid()
plt.savefig('1ºScenario - Average daily demand in a house in Winter.png')

plt.show()
```

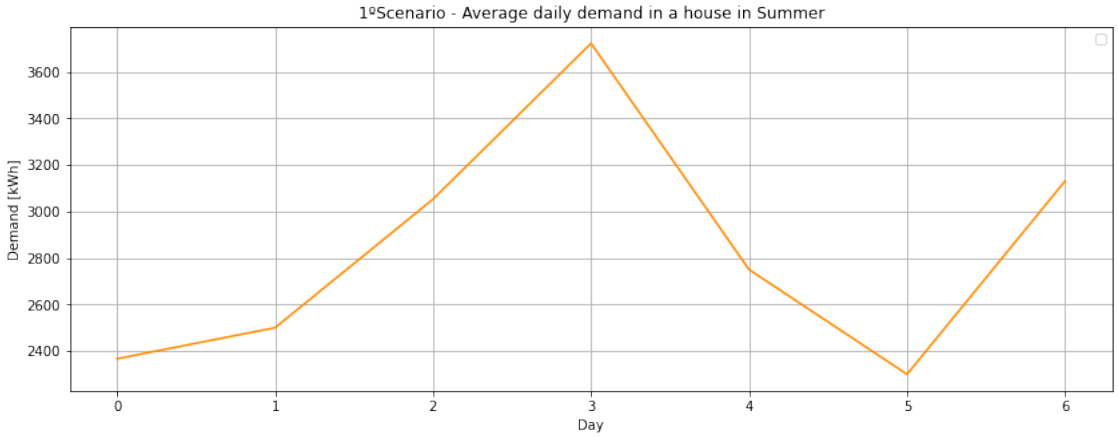
```
plt.close()
```

```
No handles with labels found to put in legend.
```



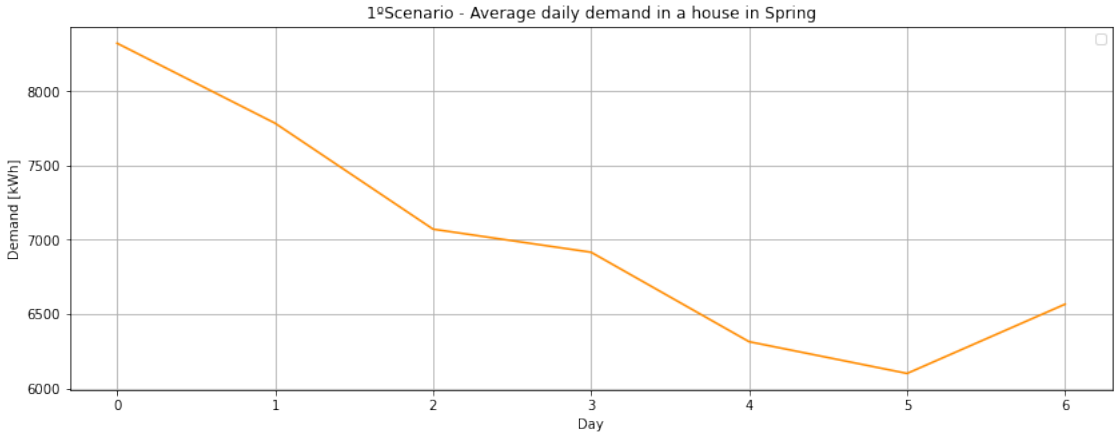
[71]:
```
plt.figure(figsize=(14,5))
plt.plot(np.arange(len(Q_medi_day_summer)), Q_medi_day_summer, 'darkorange',␣
 ↪label='')
plt.legend()
plt.title('1ºScenario - Average daily demand in a house in Summer')
plt.xlabel('Day')
plt.ylabel('Demand [kWh]')
plt.grid()
plt.savefig('1ºScenario - Average daily demand in a house in summer.png')

plt.show()
plt.close()
```

```
No handles with labels found to put in legend.
```



[72]:
```
plt.figure(figsize=(14,5))
plt.plot(np.arange(len(Q_medi_day_spring)), Q_medi_day_spring, 'darkorange',␣
 ↪label='')
plt.legend()
plt.title('1ºScenario - Average daily demand in a house in Spring')
plt.xlabel('Day')
plt.ylabel('Demand [kWh]')
plt.grid()
plt.savefig('1ºScenario - Average daily demand in a house in spring.png')

plt.show()
plt.close()
```

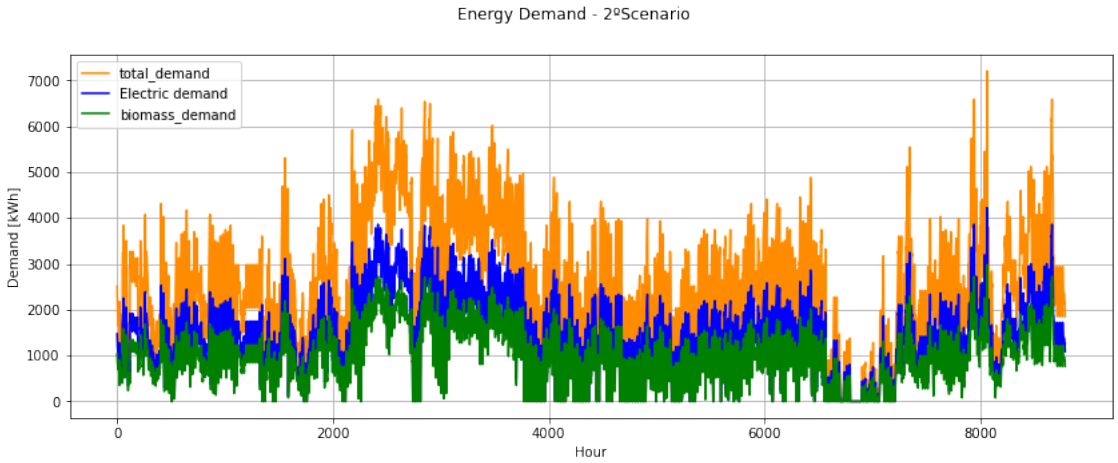
```
No handles with labels found to put in legend.
```



[114]:
```
data['Q_Island']
biomass_demand = (data['Q_Island']*0.25)/0.85
joule_heating_demand = (data['Q_Island']*0.25)
heat_pump_demand = (data['Q_Island']*0.5)/3
total_demand=biomass_demand+joule_heating_demand+heat_pump_demand
```

[118]:
```
fig, ax = plt.subplots(figsize=(14,5))
ax.plot(np.arange(len(data['Q_Island'])), total_demand, 'darkorange',␣
 ↪label='total_demand')
ax.plot(np.arange(len(data['Q_Island'])),␣
 ↪joule_heating_demand+heat_pump_demand, 'blue', label='Electric demand')

ax.plot(np.arange(len(data['Q_Island'])), biomass_demand, 'green',␣
 ↪label='biomass_demand')
```

```
ax.legend()

fig.suptitle('Energy Demand - 2ºScenario')
ax.set(xlabel='Hour', ylabel='Demand [kWh]')
ax.grid()
plt.savefig('Electric Demand hourly_2ºScenario.png')

plt.show()
plt.close()
```

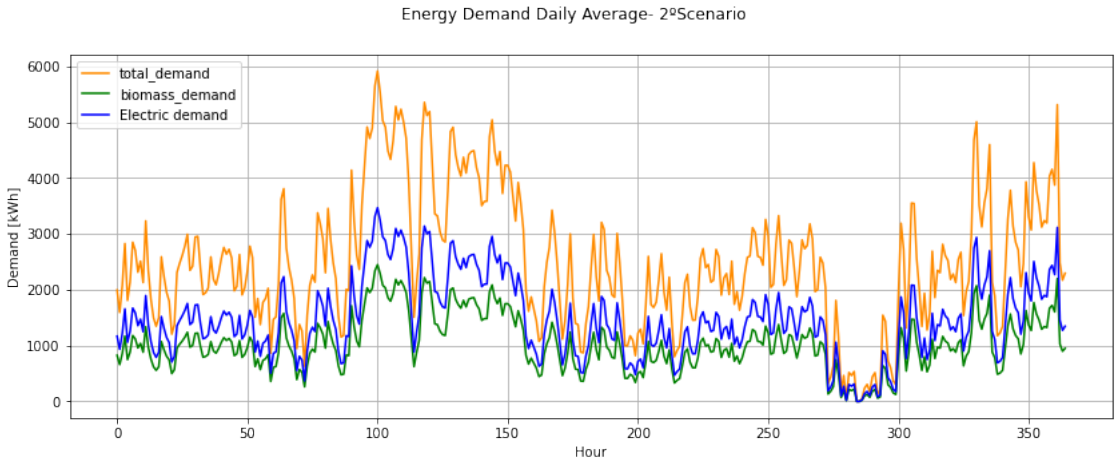


[ ]:

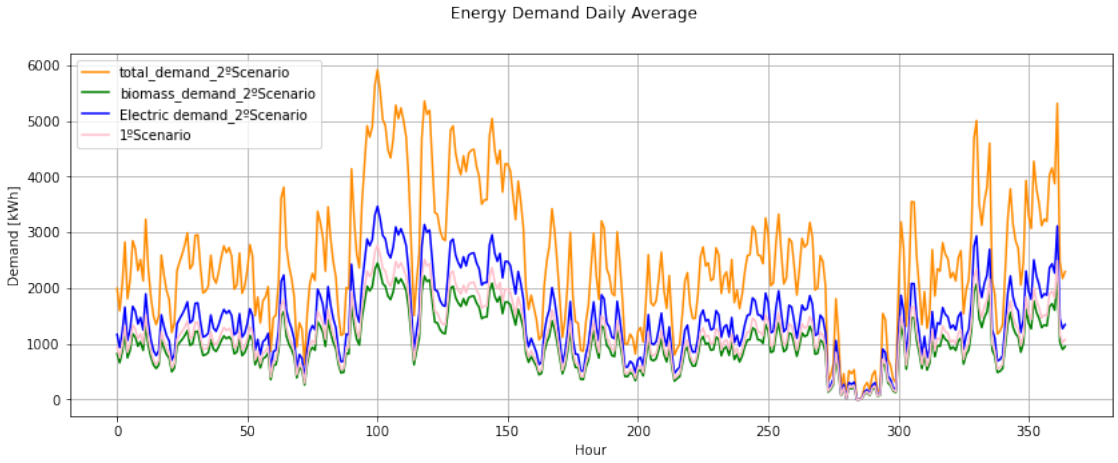
[123]:
```
Q_medi_day = []
for i in np.unique(data['J_day']):
    daily_Q = np.mean(data[data['J_day'] == i]['Q_Island'])
    Q_medi_day.append(daily_Q)
Q_medi_day = np.array(Q_medi_day)

biomass_demand = (Q_medi_day*0.25)/0.85
joule_heating_demand = (Q_medi_day*0.25)
heat_pump_demand = (Q_medi_day*0.5)/3
total_demand=biomass_demand+joule_heating_demand+heat_pump_demand
```

[124]:
```
fig, ax = plt.subplots(figsize=(14,5))
ax.plot(np.arange(len(Q_medi_day)), total_demand, 'darkorange',␣
 ↪label='total_demand')
ax.plot(np.arange(len(Q_medi_day)), biomass_demand, 'green',␣
 ↪label='biomass_demand')
ax.plot(np.arange(len(Q_medi_day)), joule_heating_demand+heat_pump_demand,␣
 ↪'blue', label='Electric demand')
```

```
ax.legend()

fig.suptitle('Energy Demand Daily Average- 2ºScenario')
ax.set(xlabel='Hour', ylabel='Demand [kWh]')
ax.grid()
plt.savefig('Electric Demand Daily Average_2ºScenario.png')

plt.show()
plt.close()
```



[128]:
```
fig, ax = plt.subplots(figsize=(14,5))
ax.plot(np.arange(len(Q_medi_day)), total_demand, 'darkorange',␣
↪label='total_demand_2ºScenario')
ax.plot(np.arange(len(Q_medi_day)), biomass_demand, 'green',␣
↪label='biomass_demand_2ºScenario')
ax.plot(np.arange(len(Q_medi_day)), joule_heating_demand+heat_pump_demand,␣
↪'blue', label='Electric demand_2ºScenario')
ax.plot(np.arange(len(Q_medi_day)), Q_medi_day_1st, 'pink', label='1ºScenario')


ax.legend()

fig.suptitle('Energy Demand Daily Average')
ax.set(xlabel='Hour', ylabel='Demand [kWh]')
ax.grid()
plt.savefig('Electric Demand Daily Average_both.png')

plt.show()
plt.close()
```


Energy Demand Daily Average
total_demand_2ºScenario
biomass_demand_2ºScenario
Electric demand_2ºScenario
1ºScenario
Demand [kWh]
Hour
0
1000
2000
3000
4000
5000
6000
0
50
100
150
200
250
300
350

[ ]: