```
#%%
%matplotlib notebook

import numpy as np
import matplotlib.pyplot as plt
import csv
import re


MonthDict = {
    1: "January",
    2: "February",
    3: "March",
    4: "April",
    5: "May",
    6: "June",
    7: "July",
    8: "August",
    9: "September",
    10: "October",
    11: "November",
    12: "December"
}
#%% md
## Die Input Klasse
Diese Klasse hat die Aufgabe, die Heizungsmessdaten aus der csv zu lesen.
#%%
class InPackage:
    def __init__(self, filename, splitter='-|:|,|T'):
        self.filename = filename
        self.splitter = splitter

    def read_and_split(self):
        with open('feeds.csv', newline='') as csvfile:
            spamreader = csv.reader(csvfile, delimiter=' ', quotechar='|')
            data = {}
            for _row in spamreader:
                try:
```

```
39                      row = re.split(self.splitter,
   _row[0])
40                      year = row[0]
41                      month = int(row[1])
42                      date = int(row[2])
43                      time = ':'.join(row[3:6])[:-3]
44                      counter = row[7]
45                      time_on = float(row[8])
46                      # entries.append(Entry(year,
   month, date, time, counter, time_on))
47                      if year not in data.keys():
48                          data[year] = dict()
49                          data[year]['t'] = 0.
50                      if month not in data[year].keys
   ():
51                          data[year][month] = {'t': 0.}
52                      if date not in data[year][month].
   keys():
53                          data[year][month][date] = {'t
   ': 0.}
54                      data[year][month][date][time] =
   time_on
55                      data[year]['t'] += time_on
56                      data[year][month]['t'] += time_on
57                      data[year][month][date]['t'] +=
   time_on
58                  except:
59                      pass
60          return data
61 #%% md
62 ## Klasse Öltank
63 #%%
64 import sympy as sp
65 import sympy.abc as abc
66
67 class Oeltank:
68     # Dimensionen:
69     lx = 3    # m
70     ly = 1.5  # m
71     lz = 0.
72
```

```
 73     def set_dh(self, h):
 74         self.lz = h
 75 
 76     def calc_v(self):
 77         return self.lx * self.ly * self.lz
 78 
 79     def get_v(self):
 80         return self.calc_v()
 81 
 82 oel = Oeltank()
 83 oel.set_dh(1.5)
 84 print(oel.get_v())
 85 #%% md
 86 # Beginn der Verarbeitung
 87 ## Einlesen der Messdaten
 88 Seit 2018 werden die Einschaltzeiten der Heizung und
     der jeweilige Ölverbrauch aufgezeichnet. Das objekt
     datareader liest mit der Methode *read_and split*
    die Messpunkte aus einer .csv .
 89 #%%
 90 datareader = InPackage(filename='feeds.csv')
 91 data = datareader.read_and_split()
 92 data
 93 #%% md
 94 ## Jahresmenge
 95 #%%
 96 fig = plt.figure()
 97 years = sorted(data.keys())
 98 plt.bar(years, [data[year]['t']/3600/24 for year in
    years])
 99 plt.title('Gemessene Heizzeit in Tage')
100 plt.grid()
101 plt.xlabel('Jahr')
102 plt.show()
103 #%% md
104 # Leistung anhand der Maschinendaten
105 Hier wird die gemessene Zeit direkt mit der
    Brennerleistung multipliziert, um zur verbrauchten
    Energie zu kommen. Später wird die Messzeit
    statistisch abgeglichen. Dabei werden Ausfalltage
    einzelner jahre mit äuivalenten Messdaten anderer
```

```
105 Jahre ersetzt.
106 Die so erhaltene wahre Zeit kann zu einer
    reevaluierung der Brennerleistung verwendet werden.
107 #%%
108 # Vergleich mit Kenndaten
109 Wirkungsgrad = 0.852
110 leistung = 70000  # WE
111 leistung = leistung / 860  # umrechnung in kW
112
113 print("Leistung aus Kenndaten: " + str(round(
    leistung, 1)) + "kW")
114 ##################################################
    ##############################
115 # Vergleich mit Ölstand
116
117 Vol_Öl = 15 * 30 * (13.7 - 6.85)  # in l
118 Heitzwert = 10.9  # kWh/l
119 Energie_Öl = Vol_Öl * Heitzwert  # kWh
120 Energie_nutz = Energie_Öl * Wirkungsgrad # kWh/l
121 #%% md
122 ## Monatsmenge
123 #%%
124 year_month_matrix = sp.zeros(len(years), 12)
125 fig = plt.figure()
126 for i, year in enumerate(years):
127     for j, month in enumerate(data[year].keys()):
128         if month == 't':
129             continue
130         m_time = data[year][month]['t']/3600
131         year_month_matrix[i-1, j-1] = m_time
132         plt.bar(month, m_time, alpha=0.2)
133 plt.title('gemessene Heizdauer in Stunden')
134 plt.xlabel('Monat')
135 plt.grid()
136 plt.show()
137 #%%
138 fig = plt.figure()
139 for i, year in enumerate(years):
140     for j, month in enumerate(data[year].keys()):
141         if month == 't':
142             continue
```

```
143         m_time = leistung * data[year][month]['t']/
    3600
144         plt.bar(month, m_time, alpha=0.2)
145 plt.title('Energiemenge in kWh')
146 plt.xlabel('Monat')
147 plt.grid()
148 plt.show()
149 #%% md
150 ## Tagesmenge
151 #%%
152 fig = plt.figure()
153 for i, year in enumerate(years):
154     for j, month in enumerate(data[year].keys()):
155         if month == 't':
156             continue
157         for day in data[year][month].keys():
158             if day == 't':
159                 continue
160             m_time = data[year][month][day]['t']/
    3600
161             plt.scatter(x=month*31+day, y=m_time,
    alpha=0.9)
162 plt.title('Heizzeit in h')
163 plt.xlabel('Tage')
164 plt.xlim((1, 365))
165 plt.grid()
166 plt.show()
167 #%%
168 fig = plt.figure()
169 for i, year in enumerate(years):
170     for j, month in enumerate(data[year].keys()):
171         if month == 't':
172             continue
173         for day in data[year][month].keys():
174             if day == 't':
175                 continue
176             m_time = leistung * data[year][month][
    day]['t']/3600
177             plt.scatter(x=month*31+day, y=m_time,
    alpha=0.9)
178
```

```
plt.title('Energiemenge in kWh')
plt.xlabel('Tage')
plt.xlim((1, 365))
plt.grid()
plt.show()
#%%
fig = plt.figure()
for i, year in enumerate(years):
    for j, month in enumerate(data[year].keys()):
        if month == 't':
            continue
        for day in data[year][month].keys():
            if day == 't':
                continue
            m_time = leistung * data[year][month][
day]['t']/3600
            plt.scatter(x=365*i+month*31+day, y=
m_time, marker='.')

plt.title('Energiemenge in kWh')
plt.xlabel('Tage')
plt.grid()
plt.show()
```