```python
#!/usr/bin/python3
from myTools import *
from ast import literal_eval

pd.option_context("max_colwidth", 1000)

#training data
precip = pd.read_csv("ECMWF_2017_2018_precip (1).csv")
precip = precip[precip.number != 0]
surface = pd.read_csv("ECMWF_2017_2018_surface.csv")
surface = surface[surface.number != 0]
surface = surface[surface.step != "0 days 00:00:00"]
res = calcWind(surface["u10"], surface["v10"])

precip.reset_index(drop=True)
surface.reset_index(drop=True)
surface["wind_direction"] = res[1].tolist()
surface["wind_speed"] = res[0].tolist()
surface["tp6"] = precip["tp6"].tolist()
allData = surface.drop(columns=["time", "step", "surface", "depthBelowLandLayer", "cape", "cin", "sd", "stl1", "swvl1", "tcc", "tcw",
                                "tcwv", "u10", "u100", "v10", "v100", "vis", "model_altitude", "model_land_usage",
                                "model_latitude", "model_longitude", "model_orography"])

allData = allData.reset_index(drop=True)
allData.t2m = allData.t2m -273.15
allData.tp6 = allData.tp6 * 1000

allData = splitInTwenty(allData)

# read and format target data
# drop columns that are of no importance for the project
syn2017 = pd.read_csv("synop_2017_March_June.csv")
syn2018 = pd.read_csv("synop_2018_March_June.csv")
synop = pd.concat([syn2017, syn2018], axis = 0)
synop = synop.iloc[1::2]
synop["precip_quantity_6hr"] = np.append(np.array([0]), accumulateTp(synop["precip_quantity_1hour"]))[:-1]
synop = synop.drop(columns=["humidity_relative", "precip_quantity_1hour", "local_datetime", "name", "lat", "lon", "community_name"])
synop = synop[synop["datetime"].str.contains("00:00:00") | synop["datetime"].str.contains("06:00:00") |
                synop["datetime"].str.contains("12:00:00") | synop["datetime"].str.contains("18:00:00")]
synop = synop.reset_index(drop=True)

#
trainingSet = createTrainingSetTimeSeries(allData, synop)
# trainingSet = createTrainingSet(allData, synop)
trainingSet.to_csv("TrainingSet.csv")

# number format from string format
trainingSet = pd.read_csv("TrainingSetDONTTOUCH2.csv")
trainingSet.dropna(inplace=True)
trainingSet = trainingSet[trainingSet.Target.str.contains("nan")==False]
trainingSet = trainingSet.reset_index()
input = trainingSet.Input.apply(literal_eval)
target = trainingSet.Target.apply(literal_eval)

# to be used later, when csv files with comlete training data have already been written (to save time)
# input = pd.read_csv("InputNotScaled.csv", header=None)
# target = pd.read_csv("TargetNotScaled.csv", header=None)

# output a table in latex format for report
to_table = pd.concat([input, target], axis=1)
with open("training_data.txt", "w") as file:
    file.write(to_table[:6].to_latex(float_format="%.3f", longtable=True))

# save data sets in csv that are readable for program (string to float through literal_eval)
input = np.array(input.values.tolist())
target = np.array(target.values.tolist())
np.savetxt("InputScaled.csv", input, delimiter=",")
np.savetxt("TargetScaled.csv", target, delimiter=",")

# if input/target in dataframe format
print (input.head())
print (target.head())
```

```
      0           1         2  ...          9         10        11
0  5.34268  250.213141  6.258264  ...  239.626345  7.293836  1.481533
1  5.34268  250.213141  6.258264  ...  165.448964  3.778853  2.362728
2  7.92130  239.626345  7.293836  ...  241.351161  7.808957  2.718449
3  5.71255  165.448964  3.778853  ...  258.250494  8.920951  0.264168
4  7.00576  241.351161  7.808957  ...  268.871264  9.503975  0.278950

    0      1      2    3
0  5.7  232.1  3.219  2.0
1  7.7  218.3  5.468  0.4
2  5.9  155.5  3.412  0.4
3  7.5  231.7  4.591  1.4
4  6.2  246.1  6.096  0.8
```