```
# imports
import pandas as pd

# generate example DataFrame
names = ["Lisa", "Sara", "Michael", "Josef"]
gender = ["f", "f", "m", "m"]
height = [164, 172, 182, 177]
# lists to DataFrame via implicitly defined python dictionary {}
data = pd.DataFrame({
        "name"      : names,
        "gender"    : gender,
        "height_cm" : height,
        })

# info
print(data)
```

```
# imports
import pandas as pd

# generate example DataFrame
names = ["Lisa", "Sara", "Michael", "Josef"]
gender = ["f", "f", "m", "m"]
height = [164, 172, 182, 177]
# lists to DataFrame via implicitly defined python dictionary {}
data = pd.DataFrame({
        "name"      : names,
        "gender"    : gender,
        "height_cm" : height,
        })

# info
print(data)


# %% plot

data.plot()


# %% bar plot

data.plot(x="name", y="height_cm", kind="bar")


# %% bar plot (Sorted)

data.sort_values(by="height_cm", ascending=False).plot(x="name", y="height_cm", kind="bar")
```

```
# imports
import pandas as pd

# generate example DataFrame
names = ["Lisa", "Sara", "Michael", "Josef"]
gender = ["f", "f", "m", "m"]
height = [164, 172, 182, 177]
# lists to DataFrame via implicitly defined python dictionary {}
data = pd.DataFrame({
        "name"      : names,
        "gender"    : gender,
        "height_cm" : height,
        })

# info
print(data)


# %% more columns

# add more
data["team"] = ["A", "B", "A", "B"]


# %% reformat table

# melt
melted = data.melt()

# info
print(melted)


# %% pivot -> smallest team member by group

pvt = data.pivot_table(
        index="gender",
        columns=None,
        values="height_cm",
        aggfunc="min"
        )

# info
print(pvt)
```

```
import seaborn as sns

# get dataset
df = sns.load_dataset('titanic')


# %% list columns

for c in df.columns:
    # show
    print(c)


# %% grouping by class

# reorder
grouping = df.groupby(by="pclass").count()["fare"]

# show
print(grouping)


# %% group and pivot

# reorder
pvt = df.pivot_table(
        index="pclass",
        columns="embarked",
        values="fare",
        aggfunc="count"
        )

# show
print(pvt)
```

```
import pandas as pd

# define filename
file = "raw_from_machine/Polysorbate40.csv"  #  - MESSED WITH

# get data
data = pd.read_csv(
            file,
            header=7,  # get column names from line number 8
            skipfooter=3,  # discard bottom 3 lines
            names=[
                "concentration_g_l",
                "surface_tension_mN_m"
                ],
            engine="python"
            )
# print data
print("Collected DATA:")
print(data)

# get "metadata" (information level)
metadata = pd.read_csv(
            file,
            skiprows=1,  # skip to line
            nrows=5,  # number of lines holding metadata
            sep=": ",  # column separator between parmeter name and
                       # value
            header=None,  # do not read column names from file
            names=["parameter", "value"],  # specify column names
                                           # explicitly
            engine="python"
            )
# print information (metadata)
print("Collected INFORMATION:")
print(metadata)



# %% plot

import matplotlib.pyplot as plt

data.plot(
    x="concentration_g_l",
    y="surface_tension_mN_m",
    color="#341256",
    marker=".",
    linestyle="--"
    )

plt.xscale("log")
```

```
# define filename
file = "raw_from_machine/Polysorbate40.csv"

# initialize results variable of type string (empty)
results = ""

# open the file for reading "r" with encoding utf-8
with open(file, "r", encoding="utf-8") as f:
    # read file line by line
    while True:
        # read line
        line = f.readline()
        # append to string
        results = results + line
        # end of file reached?
        if not line:
            print(type(line))
            # break infinite while loop
            break

# %% extract data via regex
#

# pattern as derived via "regex101.com"
pattern = "(.*),(\d+.*)"

# import python regex module
import re

# find pattern in string "results"
findings = re.findall(
                pattern,
                results
            )
# show "findings" variable (list of tuples)
print(findings)

# import pandas module to use the DataFrame with alias pd
import pandas as pd

# initialize empty pandas.DataFrame data
data = pd.DataFrame()

# initialize empty list of concentrations
concentration_g_l = []
# loop through findings to get concentrations
for _finding in findings:
    # info
    print(_finding)
    # get the first element of the "_finding" (string) and convert
    # to float
    _c_g_l = float(_finding[0])
    # append extracted concentration to list of concentrations
    concentration_g_l.append(_c_g_l)

# use concentration list as column in the defined DataFrame
data["concentration_g_l"] = concentration_g_l

# use surface tension as colums (via list comprehension)
# type conversion to float via "float()"-function
data["surface_tension_mN_m"] = [float(i[1]) for i in findings]
# print resulting DataFrame
```

```
print(data)

# clean variable space
del concentration_g_l, pattern, findings
```

```
import pandas as pd
from pathlib import Path


# define filename
file = "raw_from_machine/Polysorbate40.csv"

# read
results = pd.Series([Path(file).read_text(encoding="utf-8")])

# pattern as derived via "regex101.com"
pattern = r"(?P<concentration_g_l>.*),(?P<surface_tension_mN_m>\d+.*)"

# extract
data = results.str.extractall(pattern).reset_index(drop=True).astype(float)

# info
print(data)
```

```
import pandas as pd
from pathlib import Path
import matplotlib.pyplot as plt

# import module defined before
from surface_tension_tools import parsers

# define root directory
root = Path("raw_from_machine")

# init list
data_all = []

# loop files
for file in root.iterdir():

    # skip "bad" file
    if "MESSED" in str(file):
        continue

    # show
    print(file)

    # read
    data = parsers.get_surface_tension_data(file)

    # add source info
    data["source"] = str(file)
    data["key"] = file.stem

    # append to "all data" list
    data_all.append(data)

    # info
    print(data)

    # plot
    plt.plot(
        data["concentration_g_l"],
        data["surface_tension_mN_m"],
        label=file.stem,
        marker="."
        )

# show legend
plt.legend()
# log scale
plt.xscale("log")
# labels
plt.ylabel("surface_tension_mN_m")
plt.xlabel("concentration_g_l")


# build over df and save
data_all = pd.concat(data_all)

# save
data_all.to_excel("DATA.xlsx", index=False)
```

```
import pandas as pd
import matplotlib.pyplot as plt

# get data
data = pd.read_excel("DATA.xlsx")

# get minimum surface tension = information
info = data.groupby(by="source").min()

# show
print(info)

# %% get material information as metadata

# read
meta = pd.read_excel("Polysorbate.xlsx")

# rebuild "key"
meta["key"] = meta["Product"].str.replace(" ", "")

# show
print(meta)


# %% combine

# merge
merge = pd.merge(
            info,
            meta,
            left_on="key",
            right_on="key",
            how="outer"
            )


# %% plot

plt.scatter(
    x=merge["Molar Mass"],
    y=merge["surface_tension_mN_m"]
    )

plt.ylim(40, 50)
plt.xlabel("Molar Mass [g/mol]")
plt.ylabel("Minimum Surface Tension [mN/m]")
```