```python
#Wesley Johanson
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import matplotlib.font_manager as fm
from pylab import cm
from sklearn.linear_model import LinearRegression

print("\n\n\n")

class ChEplot:

    def __init__(self):
        self.figure=None
        self.dataLabels=None
        self.fnLabels=None
        self.numDataVars=None
        self.numDataFns=None
        self.numDataSets=None
        self.data=None
    #Data
    def loadCSV(self, filename: str, names: list, indepVars):
        # if indepVars < 1 or indepVars > len(names): return
        self.data = np.loadtxt(filename, unpack=True, delimiter=',',skiprows=0)
        # if indepVars > self.numDataSets: self.data = none; return
        self.dataLabels = names
        self.numDataVars = indepVars
        self.numDataSets = len(self.data)
        self.numDataFns = self.numDataSets - self.numDataVars

    def plotData(self, width, height):
        self.figure = plt.figure(figsize=(width, height))
        L, B, W, H = [0.15, 0.1, 0.80, 0.85]
        self.figure.axis = []
        self.figure.axis.append(self.figure.add_axes([L, B, W, H]))
        #find a way to exclude data
        for var in range(0, self.numDataVars):
            for fn in range(self.numDataVars, self.numDataSets):
                x = self.data[var]
                y = self.data[fn]
                lbl = self.fnLabels[fn - self.numDataVars]
                clr = self.dataColors[fn - self.numDataVars]
                self.figure.axis[var].plot(x,y,'.',label=lbl,color=clr)

    def segmentData(self):
        pass

    #Linear Regression
    @staticmethod
    def rSquared(x, y):
        x = x.reshape((-1,1))
        y_reg = LinearRegression().fit(x,y)
        return y_reg.score(x,y)

    def plotLRegLines(self, width=0.5, style='-', color='b'):
        for var in range(0, self.numDataVars):
```

```
58              for fn in range(self.numDataVars, self.numDataSets):
59                  m, b = np.polyfit(self.data[0],self.data[fn],1)
60                  y = (m * self.data[0]) + b
61                  self.figure.axis[var].plot(self.data[0], y, \
62                      color=self.LRegLineColors[fn-self.numDataVars], \
63                          linewidth=width ,linestyle=style)
64
65      def setLRegLineColors(self, colors=[]):
66          self.LRegLineColors = colors
67
68      def printAllRSquared(self, precision=5):
69          for fn in range(self.numDataVars, self.numDataSets):
70              for var in range(0, self.numDataVars):
71                  rSquared = ChEplot.rSquared(self.data[0],self.data[fn])
72                  rStr = "R^2 = %1." + str(precision) + "f"
73                  rStr = rStr % rSquared
74                  print( f"{rStr:<15}{self.dataLabels[fn-self.numDataVars]:<30}{'with
respect to':<20}{self.dataLabels[var]:<10}")
75
76      #Plot parameters
77      def setDataStyles(self, styles: list):      self.lineStyles = styles
78      def setDataColors(self, colors: list):  self.dataColors = colors;
self.setLRegLineColors(colors);
79      def setFnLabels(self, labels: list[str]):   self.fnLabels = labels
80
81      def setAxisLabels(self, x: str, y:str, indepVar=0, xpadding=5, ypadding=5):
82          self.figure.axis[indepVar].set_xlabel(x,labelpad=xpadding)
83          self.figure.axis[indepVar].set_ylabel(y,labelpad=ypadding)
84
85      def setTicProps(self, _size=4, _width=1, _direction='in'):
86          self.figure.axis[0].xaxis.set_tick_params(which='major', size=_size, width=_widt
direction=_direction, top='on')
87          self.figure.axis[0].xaxis.set_tick_params(which='minor', size=_size, width=_widt
direction=_direction, top='on')
88          self.figure.axis[0].yaxis.set_tick_params(which='major', size=_size, width=_widt
direction=_direction, right='on')
89          self.figure.axis[0].yaxis.set_tick_params(which='minor', size=_size, width=_widt
direction=_direction, right='on')
90
91      def setNumTics(self, delta_x=0.1, delta_y=0.1, x_subTics=3, y_subTics=3):
92          self.figure.axis[0].xaxis.set_major_locator(mpl.ticker.MultipleLocator(delta_x))
93
self.figure.axis[0].xaxis.set_minor_locator(mpl.ticker.MultipleLocator(delta_x/x_subTics
94          self.figure.axis[0].yaxis.set_major_locator(mpl.ticker.MultipleLocator(delta_y))
95
self.figure.axis[0].yaxis.set_minor_locator(mpl.ticker.MultipleLocator(delta_y/y_subTics
96
97
98      def showLegend(self, x=0.01, y=0.01, width=1, height=1, _loc='lower left',
frame=True,_fontSize=10):
99          plt.legend(bbox_to_anchor=(x,y, width, height), loc=_loc, frameon=frame,
fontsize=_fontSize)
100
101     def changeFont(self, font='Alvenir', size=10, linewidth=0.9):
102         mpl.rcParams['font.family'] = font
103         plt.rcParams['font.size'] = size
104         plt.rcParams['axes.linewidth'] = linewidth
105
106     #Presentation
```

```
107     def showPlot(self): plt.show()
108 
109     def savePlot(self, filename="poop.png", _dpi=900, _transparent=False, _bbox_inches='tight'):
110         plt.savefig(filename, dpi=_dpi, transparent=_transparent, bbox_inches=_bbox_inches)
111 
112     def close(self): plt.close(self.figure)
113 
114 
115 dataNames = ["Log_10(Reynold's Number)", "Log_10(Friction Factor)[Eqn 6]", \
116     "Log_10(Friction Factor)[Eqn15]", "Log_10(Friction Factor)[Eqn16]"]
117 fnLabels= ["Fanning $\mathcal{f}$", "$\mathcal{Re}<2\cdot10^3$", \
118     "$2100<\mathcal{Re} < 10^5$" ]
119 
120 
121 plot = ChEplot()
122 #Data
123 plot.loadCSV('logRe_logf.csv', dataNames, indepVars=1)
124 #Plotting
125 plot.setFnLabels(fnLabels)
126 plot.setDataColors(['#89CFF0','#800020','#301934'])
127 plot.plotData(width=6,height=6)
128 #Regression
129 plot.plotLRegLines(width=0.1)
130 plot.printAllRSquared()
131 #Plot Parameters
132 plot.setAxisLabels("$Log_{10}(\mathcal{Re})$", "$Log_{10}(\mathcal{f})$", xpadding=5, ypadding=5)
133 plot.setTicProps()
134 plot.setNumTics(0.1, 0.25, 3,3)
135 plot.showLegend()
136 plot.changeFont()
137 #Presentation
138 # plot.close()
139 plot.showPlot()
140 plot.savePlot(filename="log(Re)_vs_log(f).png",_dpi=600)
```