import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg  
import tkinter as tk  
from tkinter import ttk  
from tkinter import \*  
from sys import exit  
#  
# --------------------------FUNÇÕES -ALFA DIST-------------------------  
class BronstedDados:  
 def \_\_init\_\_(self,selecao):  
  
 self.biblioteca="titger.xlsx"  
 self.dados\_pKa = pd.read\_excel(self.biblioteca)  
 self.dados\_pKa.columns = self.dados\_pKa.iloc[0]  
 self.valores = self.dados\_pKa.iloc[selecao, :]  
 self.sistema = self.dados\_pKa.iloc[selecao, 0]  
 self.NH2=self.dados\_pKa.iloc[selecao, 1]  
 self.carga=self.dados\_pKa.iloc[selecao, 2]  
 self.nCOO=self.dados\_pKa.iloc[selecao, 3]  
 self.pKa=self.dados\_pKa.iloc[selecao, 4:12]  
 self.conc=0  
 print("-----",selecao,self.carga,self.valores,'-----')  
  
 def linhas(self,selecao, dados\_pKa):  
 self.valores\_linha = dados\_pKa.iloc[selecao]  
 #  
class Meio:#  
 def \_\_init\_\_(self,pH):  
 self.pKw = 14  
 self.wat=10\*\*(-pH)-10\*\*(pH-self.pKw)  
  
class Tela:  
 #  
 #dados\_pKa = pd.read\_excel('Bronsted.xlsx')  
 #sistema = dados\_pKa.iloc[:, 0]  
 #dados\_pKa.columns=dados\_pKa.iloc[0]  
 #dados\_pKa.drop(1, axis=0)  
 #lista\_dados = sistema.values.tolist()  
 #print(lista\_dados)  
   
 #  
 def \_\_init\_\_(self,master,lista\_dados=lista\_dados):  
 self.master=master  
 master.geometry("1000x600+200+50")  
 master.title(" AlfaDist ")  
 #  
 self.caixa = tk.Frame(master, borderwidth=0, relief='groove')  
 self.caixa\_botao = tk.Frame(master, borderwidth=0, relief='sunken')  
 self.caixa\_alfa = tk.Frame(master, borderwidth=5, relief='raised')  
 self.caixa\_grafico = tk.Frame(master, borderwidth=10, relief='ridge')  
 self.caixa\_pKa = tk.Frame(master, borderwidth=2, relief='solid')  
 #  
 # combo  
 #  
 n = tk.StringVar()  
 self.combo = ttk.Combobox(self.caixa, width=27, textvariable=n, state='readonly')  
 self.combo['values'] = Tela.lista\_dados  
 self.combo['state'] = 'readonly'  
 #self.combo.bind('<<ComboboxSelected>>', on\_combo)  
 self.combo.current(2)  
  
 #  
 # Entrada dos dados  
 #  
 self.caixa.grid(column=0, row=1)  
 self.texto1 = tk.Label(self.caixa, text='selecione o sistema ABBronsted')  
 self.texto2 = tk.Label(self.caixa, text='pH selecionado')  
 self.pH\_escrita = tk.Entry(self.caixa, text="3")  
  
 self.combo.grid(column=1, row=1, padx=10, pady=10)  
 self.texto1.grid(column=0, row=1, padx=10, pady=10)  
 self.texto2.grid(column=0, row=2, padx=10, pady=10)  
 self.pH\_escrita.grid(column=1, row=2, padx=10, pady=10)  
 #  
  
 # CAIXA ALFAS  
 self.caixa\_alfa.grid(column=2, row=6, padx=10, pady=10) # .place(height=500, width=50000)  
 # CAIXA GRÁFICOS  
 self.caixa\_grafico.grid(column=0, row=6, padx=10, pady=10) # .place(height=100, width=100)  
 #  
 # CAIXA pKa  
 #  
 self.caixa\_pKa.grid(column=1, row=6, padx=10, pady=10)  
 self.txt\_pKa\_titulo = tk.Label(self.caixa\_pKa, text='------ pKa ------')  
 def valor\_pH(self):  
 pH\_coleta = self.pH\_escrita.get()  
 if pH\_coleta=='':  
 pH\_coleta=3  
 print('--> pH= ',pH\_coleta)  
 return pH\_coleta  
  
 def numero\_sistema(self):  
 numero\_sistema=self.combo.current()  
 return numero\_sistema  
  
 def lista\_alfa(self):  
 pH\_coleta = float(self.valor\_pH())  
 valores=BronstedDados(Tela.numero\_sistema)  
 pKas, alfa\_valores, carga\_efetiva = alfas(valores, pH\_coleta)  
 print('\*\*\*\*\*\*', alfa\_valores, carga\_efetiva, type(pKas[1]), '\*\*\*\*\*')  
 limpa\_frame()  
 for i, item in enumerate(alfa\_valores):  
 print('\\\\', item, '////', pKas[i])  
 if i == 0:  
  
 txt\_alfa0 = tk.Label(caixa\_alfa, text='{unicodes\_value}0'.format(unicodes\_value='\u03B1'))  
 txt\_val0 = tk.Label(caixa\_alfa, text=arredonda((alfa\_valores[i])))  
 txt\_alfa0.grid(column=6, row=2, padx=10, pady=10)  
 txt\_val0.grid(column=7, row=2, padx=10, pady=10)  
  
 elif i == 1:  
  
 txt\_alfa1 = tk.Label(caixa\_alfa, text='{unicodes\_value}1'.format(unicodes\_value='\u03B1'))  
 txt\_val1 = tk.Label(caixa\_alfa, text=arredonda((alfa\_valores[i])))  
 txt\_alfa1.grid(column=6, row=3, padx=10, pady=10)  
 txt\_val1.grid(column=7, row=3, padx=10, pady=10)  
 #  
 #  
 if type(pKas[1]) == 'float':  
 txt\_pKa1 = tk.Label(caixa\_pKa, text='pKa')  
 else:  
 txt\_pKa1 = tk.Label(caixa\_pKa, text='pKa1')  
 txt\_pKa1v = tk.Label(caixa\_pKa, text=pKas[i - 1])  
 txt\_pKa1.grid(column=0, row=1, padx=10, pady=10)  
 txt\_pKa1v.grid(column=2, row=1, padx=10, pady=10)  
 #  
  
 elif i == 2:  
 txt\_alfa2 = tk.Label(caixa\_alfa, text='{unicodes\_value}2'.format(unicodes\_value='\u03B1'))  
 txt\_val2 = tk.Label(caixa\_alfa, text=arredonda((alfa\_valores[i])))  
 txt\_alfa2.grid(column=6, row=5, padx=10, pady=10)  
 txt\_val2.grid(column=7, row=5, padx=10, pady=10)  
 #  
 txt\_pKa2 = tk.Label(caixa\_pKa, text='pKa2')  
 txt\_pKa2v = tk.Label(caixa\_pKa, text=(pKas[i - 1]))  
 txt\_pKa2.grid(column=0, row=2, padx=10, pady=10)  
 txt\_pKa2v.grid(column=2, row=2, padx=10, pady=10)  
  
 elif i == 3:  
  
 txt\_alfa3 = tk.Label(caixa\_alfa, text='{unicodes\_value}3'.format(unicodes\_value='\u03B1'))  
 txt\_val3 = tk.Label(caixa\_alfa, text=arredonda(alfa\_valores[i]))  
 txt\_alfa3.grid(column=6, row=6, padx=10, pady=10)  
 txt\_val3.grid(column=7, row=6, padx=10, pady=10)  
 #  
 txt\_pKa3 = tk.Label(caixa\_pKa, text='pKa 3')  
 txt\_pKa3v = tk.Label(caixa\_pKa, text=pKas[i - 1])  
 txt\_pKa3.grid(column=0, row=3, padx=10, pady=10)  
 txt\_pKa3v.grid(column=2, row=3, padx=10, pady=10)  
  
 elif i == 4:  
 txt\_alfa4 = tk.Label(caixa\_alfa, text='{unicodes\_value}4'.format(unicodes\_value='\u03B1'))  
 txt\_val4 = tk.Label(caixa\_alfa, text=arredonda(alfa\_valores[i]))  
 txt\_alfa4.grid(column=6, row=7, padx=10, pady=10)  
 txt\_val4.grid(column=7, row=7, padx=10, pady=10)  
 #  
 txt\_pKa4 = tk.Label(caixa\_pKa, text='pKa4')  
 txt\_pKa4v = tk.Label(caixa\_pKa, text=pKas[i - 1])  
 txt\_pKa4.grid(column=0, row=4, padx=10, pady=10)  
 txt\_pKa4v.grid(column=2, row=4, padx=10, pady=10)  
  
 elif i == 5:  
 txt\_alfa5 = tk.Label(caixa\_alfa, text='{unicodes\_value}5'.format(unicodes\_value='\u03B1'))  
 txt\_val5 = tk.Label(caixa\_alfa, text=arredonda(alfa\_valores[i]))  
 txt\_alfa5.grid(column=6, row=8, padx=10, pady=10)  
 txt\_val5.grid(column=7, row=8, padx=10, pady=10)  
 #  
 txt\_pKa5 = tk.Label(caixa\_pKa, text='pKa 5')  
 txt\_pKa5v = tk.Label(caixa\_pKa, text=pKas[i - 1])  
 txt\_pKa5.grid(column=0, row=5, padx=10, pady=10)  
 txt\_pKa5v.grid(column=2, row=5, padx=10, pady=10)  
  
 elif i == 6:  
 txt\_alfa6 = tk.Label(caixa\_alfa, text='{unicodes\_value}6'.format(unicodes\_value='\u03B1'))  
 txt\_val6 = tk.Label(caixa\_alfa, text=arredonda(alfa\_valores[i]))  
 txt\_alfa6.grid(column=6, row=9, padx=10, pady=10)  
 txt\_val6.grid(column=7, row=9, padx=10, pady=10)  
 #  
 txt\_pKa6 = tk.Label(caixa\_pKa, text='pKa 6')  
 txt\_pKa6v = tk.Label(caixa\_pKa, text=pKas[i - 1])  
 txt\_pKa6.grid(column=0, row=6, padx=10, pady=10)  
 txt\_pKa6v.grid(column=2, row=6, padx=10, pady=10)  
  
 elif i == 7:  
 txt\_alfa7 = tk.Label(caixa\_alfa, text='{unicodes\_value}7'.format(unicodes\_value='\u03B1'))  
 txt\_val7 = tk.Label(caixa\_alfa, text=arredonda(alfa\_valores[i]))  
 txt\_alfa7.grid(column=6, row=10, padx=10, pady=10)  
 txt\_val7.grid(column=7, row=10, padx=10, pady=10)  
 #  
 txt\_pKa7 = tk.Label(caixa\_pKa, text='pKa 5')  
 txt\_pKa7v = tk.Label(caixa\_pKa, text=pKas[i - 1])  
 txt\_pKa7.grid(column=0, row=7, padx=10, pady=10)  
 txt\_pKa7v.grid(column=2, row=7, padx=10, pady=10)  
  
 elif i == 8:  
 txt\_alfa8 = tk.Label(caixa\_alfa, text='{unicodes\_value}8'.format(unicodes\_value='\u03B1'))  
 txt\_val8 = tk.Label(caixa\_alfa, text=arredonda(alfa\_valores[i]))  
 txt\_alfa8.grid(column=6, row=11, padx=10, pady=10)  
 txt\_val8.grid(column=7, row=11, padx=10, pady=10)  
 #  
 txt\_pKa8 = tk.Label(caixa\_pKa, text='pKa 5')  
 txt\_pKa8v = tk.Label(caixa\_pKa, text=pKas[i - 1])  
 txt\_pKa8.grid(column=0, row=8, padx=10, pady=10)  
 txt\_pKa8v.grid(column=2, row=8, padx=10, pady=10)  
  
 txt\_qef1 = tk.Label(caixa\_alfa, text='carga efetiva')  
 txt\_qef = tk.Label(caixa\_alfa, text=arredonda(carga\_efetiva))  
 txt\_qef1.grid(column=6, row=i + 4, padx=10, pady=10)  
 txt\_qef.grid(column=7, row=i + 4, padx=10, pady=10)  
 txt\_wat1 = tk.Label(caixa\_alfa, text='WAT')  
 txt\_wat = tk.Label(caixa\_alfa, text=arredonda(WAT(pH\_coleta)))  
 txt\_wat1.grid(column=6, row=i + 5, padx=10, pady=10)  
 txt\_wat.grid(column=7, row=i + 5, padx=10, pady=10)  
 txt\_pH = tk.Label(caixa\_alfa, text=' pH ----- ' + str(pH\_coleta) + ' ------- ')  
 txt\_pH.grid(column=7, row=1, padx=10, pady=10)  
  
  
  
 def coleta\_OK(self):  
 pH=self.valor\_pH()  
 numero\_sistema=self.numero\_sistema()  
 self.lista\_alfa()  
 print('------',pH,numero\_sistema,'------')  
  
 def Botoes(self,master):  
 # CAIXA BOTÕES  
 #  
 self.caixa\_botao.grid(column=0, row=2, padx=10, pady=10)  
 self.Ok\_button = tk.Button(self.caixa\_botao, text="OK", command=self.coleta\_OK)  
 self.sair\_button = tk.Button(self.caixa\_botao, text="Sair", command=master.quit)  
  
 self.Ok\_button.grid(column=1, row=5, padx=10, pady=10)  
 self.sair\_button.grid(column=3, row=5, padx=10, pady=10)  
  
  
########################################################################  
#  
#  
#  
########################################################################  
def alfas(valores\_linha,pH):  
 sistema = valores\_linha[0]  
 carga\_max = valores\_linha[2]  
 N\_pKa = valores\_linha[1] + valores\_linha[3]  
 pKas = valores\_linha[4:11]  
 pKa\_beta = []  
 pKa\_beta.append(0)  
 for i in range(0, 7):  
 if i > (N\_pKa - 1):  
 pass  
 else:  
 if pKas[i] == '':  
 print('inconsistência no número de pKas e seus valores')  
 else:  
 pKa\_beta.append((pKa\_beta[i] + pKas[i]))  
 beta\_np = np.array(pKa\_beta)  
 pH\_np = np.arange(N\_pKa + 1) \* pH  
  
 dif\_alfa = pH\_np - beta\_np  
 expo\_alfa = 10 \*\* dif\_alfa  
 alfa\_valores = expo\_alfa / (1 + (expo\_alfa.sum() - 1))  
 #  
 cargas = np.arange(start=carga\_max, stop=(carga\_max - N\_pKa - 1), step=-1)  
 carga\_efetiva=sum([cargas[i]\*alfa\_valores[i] for i in range(0,N\_pKa+1)])  
  
 return (pKas,alfa\_valores, carga\_efetiva)  
  
def tit\_AB():  
 pH = 3  
 meio = Meio(pH)  
 #  
 sistema = []  
 v\_alfa = [0, 0]  
 v\_carga = [0, 0]  
 v\_conc = [0, 0]  
 #  
 sistema1 = BronstedDados(3)  
 sistema1.conc = 0.1  
 #  
 sistema2 = BronstedDados(7)  
 sistema2.conc = 0.1  
 #  
 v\_conc[0] = sistema1.conc  
 v\_alfa[0] = sistema1.valores[1]  
 v\_alfa[1] = sistema2.valores[1]  
 print(v\_alfa[1])  
 #  
 # a,b, v\_carga[0]=alfas(v\_alfa[0],pH)  
 # a,b, v\_carga[1]=alfas(v\_alfa[1],pH)  
 for i in range(1, 2):  
 pass  
 # t=v\_carga[i]\*v\_conc[i]+Meio.wat  
 # print('---',t,'---')  
  
  
#  
#  
janela=tk.Tk()  
janela2=Tela(janela)  
janela2.Botoes(janela)  
#  
#  
#  
  
  
  
  
janela.mainloop()