```python
# coding: utf-8
__author__ = 'Geoffrey Nyaga'

# KENYA ONE PROJECT #
   # Python code to solve for CL of the wing and elliptical#
     #lift distribution without flaps .#

import sys
sys.path.append('../')
from API.db_API import write_to_db, read_from_db
from API.lifting_line_theory import llt,llt_with_plots,llt_subplots

import numpy as np
import math
import matplotlib.pyplot as plt

N = 9 # (number of segments - 1)
S = read_from_db('S') # m^2
AR = read_from_db('AR') # Aspect ratio
taper = read_from_db('taper') # Taper ratio
alpha_twist = 0 # Twist angle (deg)
i_w = 0 # wing setting angle (deg)
a_2d = 6.8754 # lift curve slope (1/rad)
alpha_0 = -4.2 # zero-lift angle of attack (deg)
clc = read_from_db('clc')

y = llt(N,S,AR,taper,alpha_twist,i_w,a_2d,alpha_0)

print(y[2])

def lifting_line_theory_combinations():

    myans = clc  #AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
    alpha_twist = np.arange(0,-4,-.01)
    # print(alpha_twist)
    i_w = np.arange(0,5,.1)

    x = []
    for i in alpha_twist:
        for j in i_w:
            lst = (i,j)
            x.append (lst)

    finalvals = []
    for alpha_twist,i_w in x:
        final = llt(N,S,AR,taper,alpha_twist,i_w,a_2d,alpha_0)
        # print (final)
        if abs( (final[2]/myans) - 1 ) <= 0.00005:
            # print ("Calculated CL",final[2],"possible combination",alpha_twist,i_w,"match")
            instant = [alpha_twist,i_w]
            finalvals.append(instant)
            llt_with_plots(N,S,AR,taper,alpha_twist,i_w,a_2d,alpha_0)
        else:
            pass
    last = np.array(finalvals).shape
    finalval = last[0]
    return finalval

lifting_line_theory_combinations()

def lifting_line_theory_subplots():

    myans = clc
    alpha_twist = np.arange(0,-4,-.01)
    i_w = np.arange(0,5,.1)

    x = []
    for i in alpha_twist:
        for j in i_w:
            lst = (i,j)
            x.append (lst)

    finalyy = []
    finalxx = []
    mycombination = []
    for alpha_twist,i_w in x:
        final = llt(N,S,AR,taper,alpha_twist,i_w,a_2d,alpha_0)

        if abs( (final[2]/myans) - 1 ) <= 0.00005:
            if __name__ == '__main__':
                print ("Calculated CL",final[2],"possible combination",alpha_twist,i_w,"match")
            mycombination.append((alpha_twist,i_w))
            myx = (final[0])
            myy = (final[1])
            finalyy.append(myy)
            yy = (np.asarray(finalyy))
        else:
            pass
    # plt.tight_layout()
    last = np.array(finalyy).shape
    finalval = last[0]
    m = math.ceil(finalval/2) #used below to define number of subplots
    fig, axes = plt.subplots(nrows=m, ncols=2)

    for ax, row in zip(axes.flatten(), finalyy):
      ax.plot(myx,row,'r-'  )
      # ax.set_label('Label via method')
      # ax.legend()
    # turn remaining axes off
    for i in range(len(finalyy),m):
      axes.flatten()[i].axis("off")
      # ax.legend()
      # ax.title(i)
    plt.tight_layout()
    if __name__ == '__main__':
        plt.show()
    return mycombination

x = lifting_line_theory_subplots()
def final_subplot():
    num = 1
    # num = int(input('Select the graph:'))   #AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
    return x[num-1]

y = final_subplot()
if __name__ == '__main__':
    print(y[0],"this is twist",y[1],"and this is the wing incidence")
write_to_db('alpha_twist',y[0])
write_to_db('wing_incidence',y[1])

```