Nama : Andri Firman Saputra  
NIM : 201011402125  
Kelas : 06TPLP016  
Link GitHub : <https://github.com/hako-975/uas_kecerdasan_buatan>

import numpy as np

from matplotlib import pyplot as plt

class BaseFuzzy():

    def \_\_init\_\_(self):

        self.minimum = 0

        self.maximum = 0

    def up(self, x):

        return (x - self.minimum) / (self.maximum - self.minimum)

    def down(self, x):

        return (self.maximum - x) / (self.maximum - self.minimum)

class Temp(BaseFuzzy):

    def \_\_init\_\_(self):

        self.t1 = 0

        self.t2 = 40

        self.t3 = 60

        self.t4 = 80

        self.tn = 100

    def freeze(self, x):

        if x < self.t1:

            return 1

        elif self.t1 <= x <= self.t2:

            self.minimum = self.t1

            self.maximum = self.t2

            return self.down(x)

        else:

            return 0

    def cold(self, x):

        if self.t1 <= x <= self.t2:

            self.minimum = self.t1

            self.maximum = self.t2

            return self.up(x)

        elif self.t2 <= x <= self.t3:

            self.minimum = self.t2

            self.maximum = self.t3

            return self.down(x)

        else:

            return 0

    def warm(self, x):

        if self.t2 <= x <= self.t3:

            self.minimum = self.t2

            self.maximum = self.t3

            return self.up(x)

        elif self.t3 <= x <= self.t4:

            self.minimum = self.t3

            self.maximum = self.t4

            return self.down(x)

        else:

            return 0

    def hot(self, x):

        if self.t3 <= x <= self.t4:

            self.minimum = self.t3

            self.maximum = self.t4

            return self.up(x)

        elif x > self.t4:

            return 1

        else:

            return 0

class Pressure(BaseFuzzy):

    def \_\_init\_\_(self):

        self.p1 = 0.0

        self.p2 = 0.2

        self.p3 = 0.4

        self.p4 = 0.6

        self.p5 = 0.8

        self.p6 = 1.0

    def very\_low(self, x):

        if x <= self.p2:

            return 1

        elif self.p2 < x <= self.p3:

            self.minimum = self.p2

            self.maximum = self.p3

            return self.down(x)

        else:

            return 0

    def low(self, x):

        if self.p2 <= x <= self.p3:

            self.minimum = self.p2

            self.maximum = self.p3

            return self.up(x)

        elif self.p3 < x <= self.p4:

            self.minimum = self.p3

            self.maximum = self.p4

            return self.down(x)

        else:

            return 0

    def medium(self, x):

        if self.p3 <= x <= self.p4:

            self.minimum = self.p3

            self.maximum = self.p4

            return self.up(x)

        elif self.p4 < x <= self.p5:

            return 1

        else:

            return 0

    def high(self, x):

        if self.p4 <= x <= self.p5:

            self.minimum = self.p4

            self.maximum = self.p5

            return self.up(x)

        elif self.p5 < x <= self.p6:

            self.minimum = self.p5

            self.maximum = self.p6

            return self.down(x)

        else:

            return 0

    def very\_high(self, x):

        if x >= self.p6:

            return 1

        elif self.p5 < x < self.p6:

            self.minimum = self.p5

            self.maximum = self.p6

            return self.up(x)

        else:

            return 0

class Speed(BaseFuzzy):

    def \_\_init\_\_(self):

        self.slow = [0, 40, 60]

        self.steady = [40, 60, 80, 100]

        self.fast = [80, 100, 100]

    def calculate\_speed(self, temperature, pressure):

        if temperature == 'FREEZE' and pressure == 'VERY LOW':

            return self.fast

        elif temperature == 'COLD' and pressure == 'VERY LOW':

            return self.fast

        elif temperature == 'WARM' and pressure == 'VERY LOW':

            return self.fast

        elif temperature == 'HOT' and pressure == 'VERY LOW':

            return self.fast

        elif temperature == 'FREEZE' and pressure == 'LOW':

            return self.fast

        elif temperature == 'COLD' and pressure == 'LOW':

            return self.steady

        elif temperature == 'WARM' and pressure == 'LOW':

            return self.steady

        elif temperature == 'HOT' and pressure == 'LOW':

            return self.steady

        elif temperature == 'FREEZE' and pressure == 'MEDIUM':

            return self.steady

        elif temperature == 'COLD' and pressure == 'MEDIUM':

            return self.steady

        elif temperature == 'WARM' and pressure == 'MEDIUM':

            return self.steady

        elif temperature == 'HOT' and pressure == 'MEDIUM':

            return self.steady

        elif temperature == 'FREEZE' and pressure == 'HIGH':

            return self.steady

        elif temperature == 'COLD' and pressure == 'HIGH':

            return self.steady

        elif temperature == 'WARM' and pressure == 'HIGH':

            return self.steady

        elif temperature == 'HOT' and pressure == 'HIGH':

            return self.slow

        elif temperature == 'FREEZE' and pressure == 'VERY HIGH':

            return self.slow

        elif temperature == 'COLD' and pressure == 'VERY HIGH':

            return self.slow

        elif temperature == 'WARM' and pressure == 'VERY HIGH':

            return self.slow

        elif temperature == 'HOT' and pressure == 'VERY HIGH':

            return self.slow

    def graph(self, temperature, pressure):

        x = np.linspace(-10, 110, 1000)

        slow\_membership = np.array([self.slow[0], self.slow[0], self.slow[1], self.slow[2]])

        steady\_membership = np.array([self.steady[0], self.steady[1], self.steady[2], self.steady[3]])

        fast\_membership = np.array([self.fast[0], self.fast[1], self.fast[2], self.fast[2]])

        slow\_values = np.array([self.membership\_function(slow\_membership, value) for value in x])

        steady\_values = np.array([self.membership\_function(steady\_membership, value) for value in x])

        fast\_values = np.array([self.membership\_function(fast\_membership, value) for value in x])

        plt.figure(figsize=(10, 6))

        plt.plot(x, slow\_values, label='Slow')

        plt.plot(x, steady\_values, label='Steady')

        plt.plot(x, fast\_values, label='Fast')

        plt.title('Speed Output for Temperature: ' + temperature + ' and Pressure: ' + pressure)

        plt.legend()

        plt.show()

    def membership\_function(self, membership, x):

        if x <= membership[0] or x >= membership[-1]:

            return 0

        elif membership[0] < x < membership[1]:

            return (x - membership[0]) / (membership[1] - membership[0])

        elif membership[1] <= x <= membership[2]:

            return 1

        elif membership[2] < x < membership[3]:

            return (membership[3] - x) / (membership[3] - membership[2])

        else:

            return 0

speed = Speed()

temperature = input("Enter temperature: ")

pressure = input("Enter pressure: ")

speed\_values = speed.calculate\_speed(temperature.upper(), pressure.upper())

speed.graph(temperature.upper(), pressure.upper())