```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
from vaccination import Vaccination
class FuzzSystem():
    #Constructor Method
    def __init__(self):
        #Inspiried from these websites:

#https://pythonhosted.org/scikit-fuzzy/auto_examples/plot_tipping_problem_newapi.html

#https://pythonhosted.org/scikit-fuzzy/auto_examples/plot_control_system_advanced.html#example-plot-control-system-advanced-py
        # New Antecedent/Consequent objects hold universe variables and membership
        # functions
        self.vacc = ctrl.Antecedent(np.arange(0,1.01, 0.01), 'vacc_rates')

        self.control = ctrl.Consequent(np.arange(-0.2,0.21, 0.05), 'control_rates')

        # Custom membership functions can be built interactively with a familiar,
        # Pythonic API
        #  Vacc Set Partioning
        self.vacc['low'] = fuzz.trapmf(self.vacc.universe, [0, 0, 0.4, 0.6])
        self.vacc['medium'] = fuzz.trimf(self.vacc.universe,[0.4, 0.6, 0.8])
        self.vacc['high'] =  fuzz.trapmf(self.vacc.universe, [0.6, 0.8, 1, 1])
        # Printing set partioning
        self.vacc.view()
        # Custom membership functions can be built interactively with a familiar,
        # Pythonic API
        # Control Set Partioning
        self.control['low'] = fuzz.trapmf(self.control.universe, [-0.2, -0.2, -0.1,0])
        self.control['medium'] = fuzz.trimf(self.control.universe, [-0.1, 0, 0.1])
        self.control['high'] = fuzz.trapmf(self.control.universe,[0, 0.1, 0.2,0.2])
        # Printing set partioning
        self.control.view()

        #---------------------------------------#
        #-----------------RULES-----------------#
        #---------------------------------------#
        #0) If vacc rate is low,-----------------#
        #-------control output will be high-----#
        #---------------------------------------#
        #1) If vacc rate is mid,-----------------#
        #-------control output will be mid------#
        #---------------------------------------#
        #2) If vacc rate is high,----------------#
        #-------control output will be low------#
        #---------------------------------------#
        rule0 = ctrl.Rule(antecedent=self.vacc['low'] ,
                          consequent=self.control['high'], label='rule high')

        rule1 = ctrl.Rule(antecedent= self.vacc['medium'],
                          consequent=self.control['medium'], label='rule mid')

        rule2 = ctrl.Rule(antecedent= self.vacc['high'],
                          consequent=self.control['low'], label='rule low')

        #Apply Rules
        self.controlRules = ctrl.ControlSystem(rules=[rule0, rule1, rule2])
        #Initialize system
        self.system=Vaccination()

    #This method is going to apply fuzzy logic to current system in one iteration
    def applyFuzzLogic(self):
        #Apply simulation
        self.fuzz = ctrl.ControlSystemSimulation(self.controlRules)
        #Get current vacc rate
        vacc_rate = self.getVaccRate()
        #Set input as vacc rate
        # Pass inputs to the ControlSystem using Antecedent labels with Pythonic API
        self.fuzz.input['vacc_rates'] = vacc_rate
        #This compute will compute the output (defuzzy)
        self.fuzz.compute()
        #Get output
```

```
        outputControl=self.fuzz.output['control_rates']
        #Apply this control output value
        self.system.vaccinatePeople(outputControl)

    #Getting the last element of vacc rate curve
    def getLastRate(self):
        return self.system.vaccination_rate_curve_[-1]
    #Getting the current vacc rate
    def getVaccRate(self):
        return self.system.checkVaccinationStatus()[0]
    #Getting the current fail rate
    def getFailRate(self):
        return self.system.checkVaccinationStatus()[1]



berkay=FuzzSystem() #Main system
flag=True #Flag will be True until equilibrium point
cost=0      #Initial Cost value
stopDiff=0.00001 #Stopper diff
for slot in range(200):
    prev_vacc_rate=berkay.getVaccRate() #Get previous vacc rate
    berkay.applyFuzzLogic() #Apply the control by calling the method
    vacc_rate =berkay.getVaccRate() #Get current vacc rate
    if(flag==True):
        cost+=berkay.getLastRate()#Sum all costs until equilibrium point
    currentDiff=abs(vacc_rate-prev_vacc_rate) #Calculate the diff between percentages of
consc two iteration
    if(currentDiff< stopDiff ) and flag==True: #Check if the difference is enough small
        flag=False #Flag will be Flase when equilibrium point is reached
        point_ss=slot #record the time when there is a equibilirium

berkay.system.viewVaccination(point_ss = point_ss, vaccination_cost = cost,
filename='vacc-v1')
```