```python
"""
Created on Mon Nov  9 19:11:02 2020

@author: Sivaraman Sivaraj
"""

import numpy as np
import math

def f(x): #for example, we use this square function
    value = 0
    for i in range(len(x)):
        value += (x[i]**2)
    return value

def Fit_function(F,N):
    Fitness_Value = list()# to store fitness function value
    for j in range(len(N)):
        temp = F(N[j])
        Fitness_Value.append(temp) # storing the function of each particle values
    return Fitness_Value

### To check Function #####
# d= np.array([[1,2,3,4],[1,2,3,7]])
# print((Fit_function(f, d)))
#### ##########

def corner_bounding(p,lb,ub):
    """
    Parameters
    ----------
    p : particle's vector
    lb : lower bound
    ub : upper bound
    Returns
    -------
    bounded vector

    """
    bounded_vector = list()
    for i in range(len(p)):
        if lb <= p[i] and p[i] <=ub:
            bounded_vector.append(p[i])
        elif p[i] > ub:
            bounded_vector.append(ub)
        elif p[i] < lb:
            bounded_vector.append(lb)
    return bounded_vector

### To check corner bounding #####
# ss = [-2,-34,2,3,4,5,6,4,2,3,123,10,323]
# print(corner_bounding(ss, -5, 10))
###############
```

```python
def Do_PSO(N,T,w,c1,c2,F,lb,ub):
    """
    Parameters
    ----------
    N : Input particles and it'svector
    T : Number of iteration
    w : inertia weight
    c1 : cognitive term constant
    c2 : social term constant
    F : Fitness function(where we can define our own)
    lb : Lower Bound of Decision Vector
    ub : Upper Bound of Decision Vector

    Governing Equation:
        Xi = Xi + Vi
        Vi = wVi + c1*r1*(P_best - Xi) + c2*r2*(g_best - Xi)

    Returns
    -------
    global_vector,global_best,N_updated, F_updated,N
    """

    Fitness_Value = Fit_function(F, N) # Finding intial fitness values


    local_best = Fitness_Value[::]# at start, to update best local value after each iterat
ion
    global_best = min(local_best) #maximization or minimization depending on need
    global_vector = N[Fitness_Value.index(global_best)]

    P_best_local_vector = N[::]# local best vector for each particle, at inital as it is
    local_best = Fitness_Value[::]

    N_updated  = [] #to store the particle's position after each iteration
    N_updated.append(N)#just to consider intial position
    F_updated  = [] ##to store the particle's function values after each iteration
    F_updated.append(Fitness_Value)#just to consider intial value


    for itr in range(T):
        V = np.random.randint(lb,ub,np.shape(N)) # rvd = random_velocity_vector, initiate
random value
        for i in range(len(N)):
            Xi = np.array(N[i])
            Vi = V[i]
            P_best_vector = P_best_local_vector[i]
            r1 = np.random.rand(len(N[i]))
            r2 = np.random.rand(len(N[i]))
            Vi_next = (w*Vi) + c1*r1*(P_best_vector - Xi) + c2*r2*(global_vector - Xi)#gov
erning function
            Xi_new = Xi + Vi_next #updating the equation

            Xi_new = corner_bounding(Xi_new,lb,ub) #corner bounding the iterated value
```

```
            N[i] = Xi_new# updating new position vector


        F_val = Fit_function(F, N)

        if min(F_val) < global_best: #updating the global vector
            global_best = min(F_val)
            global_vector = N[F_val.index(global_best)]

        for p in range(len(F_val)): #updating the local best vector
            if F_val[p] < local_best[p]:
                local_best[p] = F_val[p]
                P_best_local_vector[p] = N[p]


        N_updated.append(N)
        F_updated.append(F)
        print(itr)


    return global_vector,global_best,N_updated, F_updated,N


#### to check the code###
# N1 = np.random.randint(1,10,(5,4))
# a,b,c,d,e = Do_PSO(N1.tolist(),15,0.9,1.5,1.6,f,2,8)
# print(c)
######### END ##########

def constriction_coefficients(k,q1,q2):
    q = q1+q2
    if k<=1 and k >=0 and q >4:
        kai = (2*k)/abs(2-q-(math.sqrt((q**2)-(4*q))))
        return kai, kai*q1, kai*q2
    else:
        return "choose different values for constraint satisfaction"
```