```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed May  8 13:14:59 2019

@author: hernan
EA Seminar

Problems classes and benchmark functions

Multi-objective
- Multi-objective knapsack Problem MKP

"""

# Problem

class MKP:
    """ Multi-objective Knapsack Problem
    http://home.ku.edu.tr/~moolibrary/
    Examples:
        MKP/KP_p-2_n-10_ins-1.dat
        n-10, n20, ..., n100
        ins-1, ins2, ..., ins-10

    File format
        Number of objective functions (p)
        Number of items (n)
        Capacity of the knapsack (W∈ℤ)
        Profits of the objects in each objective function, ℤp×n  [[],[]]
        Weights of the objects (w∈ℤn) []

    """

    def __init__(self, fname):
        """Constructor

        Parameters
        ----------
        fname: string
            file name where the problem is described
        """

        self.fname = fname
        fproblem = open(self.fname)
        # read nobj, nitems, capacity
        self.nobj = int(fproblem.readline())
        self.nitems = int(fproblem.readline())
        self.capacity = int(fproblem.readline())
        # read profits of nitems for each objective
        self.profit=[]
        bad_chars = '[],'
        for k in range(self.nobj):
            line = (fproblem.readline())
            line = "".join(c for c in line if c not in bad_chars)
            line = line.split()
            pi = list(map(int,line))
            self.profit.append(pi)
        #print (self.profit)

        # read weigth of nitems
        line = (fproblem.readline())
        line = "".join(c for c in line if c not in bad_chars)
        line = line.split()
        self.weigth = list(map(int,line))
        #print (self.weigth)

        fproblem.close()

    def fitness(self, x):
        """Fitness function

        Parameters
        ----------
        x: list
            Variables

        Returns
        -------
        Tuple
            Fitness values
        """

        f = [0.0]*self.nobj
        #print(f)
        for k in range(self.nobj):
            for i in range(len(x)):
               # print(k,i)
                f[k] = f[k] + self.profit[k][i]*x[i]
        g = 0
        for i in range(len(x)):
            g = g + self.weigth[i]*x[i]

        # penalization term if
        if g > self.capacity:
            f = [-(g-self.capacity)  for fi in f]
#               f = [fi -(g-self.capacity)  for fi in f]
        return tuple(f)
```