Dynamic Pricing Competition
Scenario descriptions

Scenario 2: Oligopoly with multiple products

In this scenario, participants will compete in markets consisting of three to six players, where each player in the market sells $J = 3$ different products with unbounded inventory. We run a large number of S simulations. In each simulation we randomly select three to six players from all participants to compete against each other during $T = 1000$ discrete time periods indexed by $t = 1, \ldots, T$. Let n denote the total number of players in a simulation, and let the selected players in this simulation be indexed by $i = 1, \ldots, n$. In each time period $t \in \{1, \ldots, T\}$ all players $i = 1, \ldots, n$ determine a price $p_{i,j,t} \in \{0.00, 0.01, \ldots\}$ for each of their J products $j = 1, \ldots, J$, after which sales $s_{i,j,t}$ is realized, for each player $i = 1, \ldots, n$ and product $j = 1, \ldots, J$. Sales is realized according to an undisclosed sales-generating mechanism that may be different in each simulation (but, conditionally on selling prices, statistically identical in different time periods within the same simulation). The prices are observable for all players, but each player can *only observe his/her own sales.* After sales has realized, each player $i = 1, \ldots, n$ earns revenue $\sum_{j=1}^{J} p_{i,j,t} s_{i,j,t}$ and one proceeds to the next time period. The objective is to maximize expected revenue earned over the whole time horizon $t = 1, \ldots, T$, averaged over all simulations in which one has participated. The following domain knowledge is available: (i) sales is stationary (i.e. within a simulation, two time periods t_1, t_2 with the same prices $p_{i,j,t_1} = p_{i,j,t_2}$ for all i and j generate the same sales distribution), (ii) the distribution of sales $s_{i,j,t}$ in any period t may depend on *all* prices $(p_{i',j',t})_{1 \leq i' \leq n,\, 1 \leq j' \leq J}$ and (iii) prices larger than 100 will generate a negligible amount of sales, regardless the prices of other players and products).

Sample code Scenario 2:

```
import numpy as np

def p(prices_historical=None, demand_historical=None, information_dump=None):
    """
    this pricing algorithm would return the minimum price for the product
    used by any competitor in the last iteration, it returns a random
    price if it is the first iteration

    input:
        prices_historical: numpy 3-dim array: (number competitors) x
           (number of products = 3) x (past iterations)
                           it contains the past prices of each competitor
                           (you are at index 0) over the past iterations
        demand_historical: numpy 2-dim array: (number of products = 3) x
         (past iterations)
                           it contains the history of your own past observed demand
                           over the last iterations
        information_dump: some information object you like to pass to yourself
                          at the next iteration
    """

    if demand_historical is None :
        return ( np.random.uniform(30,80,3).round(1) , None)

    next_price = [  np.min(prices_historical[1:,0,-1:]).round(1),
                    np.min(prices_historical[1:,1,-1:]).round(1),
                    np.min(prices_historical[1:,2,-1:].round(1))  ]

    return ( next_price , information_dump )
```