Homework 06
Prisoner’s dilemma

BE5B33PRG – Programming Essentials

Prisoner's dilemma

- Two people questioned about the same crime
- Talking to the interrogator separately
- Both given the same deal: Vouch for the other person's innocence or guilt

Person A claims B is…	Person B claims A is…	Person A – outcome	Person B - outcome
Innocent	Innocent	3-month reduction	3-month reduction
Innocent	Guilty	0-month reduction	5-month reduction
Guilty	Innocent	5-month reduction	0-month reduction
Guilty	Guilty	1-month reduction	1-month reduction

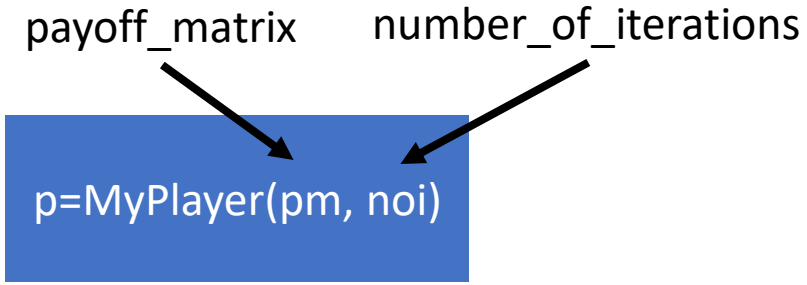
Strategies

- Made up of two moves – Cooperate (“B is innocent”) and Defect (“B is guilty”)
- Points instead of months off sentence
- Several (many) rounds

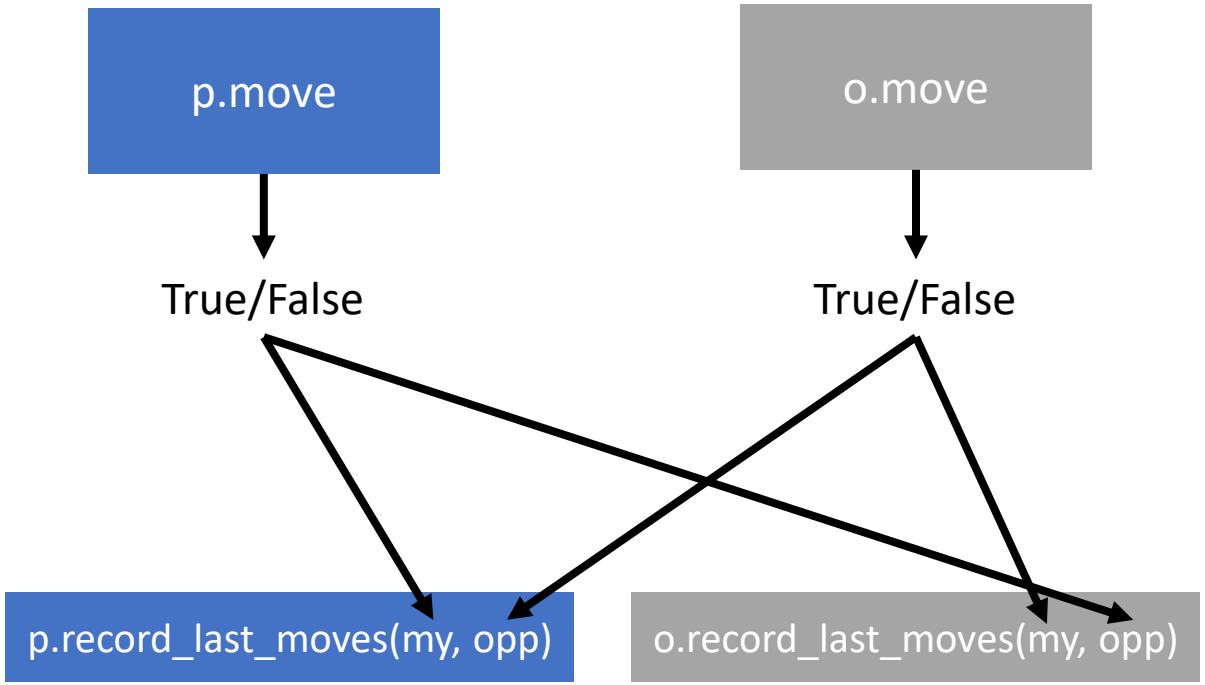
- Strategy based on past interactions with that player
- Basic strategies:
  - Always cooperate
  - Always defect
  - Cooperate unless the other player defects
  - Figure out opponent’s strategy and play what’s best against that
- Advanced strategies

Player specification

- Implement MyPlayer class



Method	Inputs	Return values
__init__	• Payoff_matrix (2x2 matrix of tuples) • Number_of_iterations (optional)	None
move	None	False (COOPERATE, stay silent)/True (DEFECT, betray)
record_last_moves	• my_last_move (True/False) • opponent_last_move (True/False)	None



Payoff matrix

- Assumption: Always symmetric

player's A move (index 0)		player's B move (index 1)
		prisoner B stays silent (COOPERATEs) [?][0][?]	prisoner B betrays (DEFECTs) [?][1][?]
prisoner A stays silent (COOPERATEs) [0][?][?]	(A's reward [0][0][0]: 4, B's reward [0][0][1]: 4)	(A's reward [0][1][0]: 1 B's reward [0][1][1], 6)
prisoner A betrays (DEFECTs) [1][?][?]	(A's reward [1][0][0]: 6, B's reward [1][0][1]: 1)	(A's reward [1][1][0]: 2, B's reward [1][1][1]: 2)

```
payoff_matrix = ( ((4,4),(1,6)) , ((6,1),(2,2)) )
```

It is forbidden…

- to write to a hard-drive. All you need to store in-between move, store in the (RAM) memory.
- to connect to other computers, processes, etc.
- to search the hard-drive
- to dishonourable influence opponent's decision making
- …

Simply put, the player's job is to play!

Submission #1 (06_pd)

- Submit at least the most basic (but working!) player
- Checklist (2 points):
  - Module called player.py (only and only this file)
  - Non-empty docstring.
  - Possible to create an instance of the MyPlayer class with 1 input parameter (payoff matrix)
  - Possible to create an instance of the MyPlayer class with 2 input parameters (payoff matrix, number of iterations)
  - The move() method returns True or False (Boolean logic type)
  - The record_last_moves() exists and it is capable of accepting input parameters.
- Another 2 points for creativity and well-written code.

Submission #2 (06_tour)

- Tournament!
- Maximize total score = sum of all partial gains from all rounds in all matches (against all other players)
- Two tournaments:
  - **Basic version** - the payoff matrix, as well as the number of iterations will be known to the student beforehand.
  - **Advanced version** - the payoff matrix will be passed to the player just before the beginning of the game. The information about the number of iteration may or may not be passed to the player.
- Evaluation:
  - Player with memory – Up to 4 points
  - Ranking in tournament #1 – 0.5 to 2 points
  - Ranking in tournament #2 – 0.5 to 2 points