Tournament

Main info

I am organizing a game AI tournament inspired by the famous Axelrod's tournament in Prisoner's dilemma from 1970s.

We will play a repeated symmetrical bimatrix game. There are four available strategies (A, B, C, D) in the game. The payoffs are defined by the following bimatrix.

\	A	B	C	D
A	6 \ 6	3 \ 1	3 \ 1	3 \ 1
B	1 \ 3	2 \ 2	8 \ 0	8 \ 0
C	1 \ 3	0 \ 8	10 \ 10	1 \ 18
D	1 \ 3	0 \ 8	18 \ 1	4 \ 4

For example, if you choose A and you opponent chooses B, you will obtain 3 points and your opponent will obtain 1 point. Note that if we restrict ourselves only to strategies C and D, the game becomes equivalent to the Prisoner's dilemma (see the bottom right 2x2 submatrix).

Your task is to create a script in Python (required version Python 3.6 or higher) which will play the game.

Implementation

Repository: https://github.com/sauermar/Turnaj-M-M

The rules are simple. Before we go through them with you, download the source codes from the repository and look into them while reading this guide.

First of all, your script has to implement an interface `Player` from the file `player.py`. It means that your script must meet the requirements of what specific methods must return and what arguments they are given, as described in the interface. In Python, a method is a function that takes the object itself as the first argument (`self`).

If you want to make the implementation process easier for yourself, you can copy the sample module `mirror.py` (which already implements the interface) and easily modify the methods there.

The interface specifies an initializer `__init__`. This method is always called right after the object is created. The player object will be created at the beginning of every interaction (a combat between two players). The initializer is suitable for creating attributes, which are kind of local variables inside your object. They can e.g. store the previous state of the game.

The method `author_name` should be modified to return your (full) name as a string.

The method `next_move` chooses the player's next move using the type `Move`, which describes available moves (strategies). They correspond to the rows and the columns of the bimatrix.

```
class Move(enum.Enum):
    a_safe_way = ("A", 0)
    betray = ("B", 1)
    cooperate = ("C", 2)
    deceive = ("D", 3)
```

In order to use this `enum`, you need to import `player.py` into your script. It can be done by writing `from player import Move` at the beginning of your code. Then you can use it either by its name `Move.cooperate` or by its value `Move(("C", 2))`.

After each game turn, you can obtains results via the method `reward`. The class `Result` (declared in `result.py`) contains your move, opponent's move and obtained points. They are accessible using methods `get_my_score` and `get_opp_score`.

At this point, you know everything you need to know in order to write your own script. Its development is up to you! In case you need a hint, there are a few scripts which you can have a look at. They are stored in the following files: `always_cooperate.py`, `always_deceive.py`, `unforgiving.py`, `score_counting.py`, `mirror.py`

In order to test your implementation, you can play against your player manually using the program `testing.py` where you must specify the name of your script dot the name of your class (the one that implements the interface `Player`) as the first argument, and the iteration count as the (optional) second argument. Example call: "`python testing.py mirror.Mirror 150`" The default iteration count is 10. Please, keep in mind that your script must be in the same directory where the file `testing.py` is located.

Finally, when you are satisfied with your script, send me your source code via e-mail.

Tournament details

All interactions (combats) in the tournament will have the same number of rounds. We can't tell you the number beforehand and it will be a different value for each tournament, but you can be sure that it will be an integer between 10 and 1000.

There will be three kinds of players in the tournament.

- Competitors of M&M (something like a math competition in Czechia)
- Other participants (like you)
- Fixed 19 default scripts (you can see them in the folder `trivial`)

Human players will have 3 instances each. Default (trivial) players will have 1 instance each. Those instances will play all against all (a.k.a. round-robin tournament) and the total sum of one's points from all combats will matter. For the final results, the median score of your three instances will be counted.

This implies that each instance of your script will interact three times with each script written by other competitors, twice with itself, and once with each default script.

Privacy

Complete detailed results (including your full name) will be shared via our mailing list `hry@mam.mff.cuni.cz`, which is private. Only organizers will have access to your source codes. The official webpage of the M&M competition will display results and names only of M&M competitors, who have submitted an explicit GDPR-compliant consent. No information about other participants of the tournament (like you) will be publicly available.

If you want to ask about anything, drop me a line!
martin.dvorak@ist.ac.at (organizer)
market.sauer@gmail.com (author of the original source code)