Welcome to Antwarz!

Control your ant colony and defeat opponents in this programming challenge.

Welcome to Antwars!

Introduction

Welcome to antwars! In this challenge, you will have to create bots to control an ant colony and defeat your opponent.
If you are not familiar with certain algorithms and/or datastructures, this challenges could be time consuming. If you are here for a pure cybersecurity CTF experience, you might want to check out other challenges. If however you are not afraid of a little programming challenge, you have come to the right place.
You will first need to prove yourself by facing two of my own bots in order to deserve you first two flags. Once you have proved your are worthy of the Arena, you will be allowed to face other players!
Please make sure to report any security fault you might find, as it's not the goal of the challenge to exploit the platform. Your code is ran in a sandbox. If you find any python builtins missing that you require, please request it in a ticket.

Submissions

There are two types of submissions.

1. **Test submissions**: A game submitted through this kind of submission can be reviewed afterwards, but does not win a flag in case of victory.
2. **Submissions**: It will run 15 games, in which at least 12 need to be won by the player. This will award the player a flag, and unlock the next arena (upon flag submission on the CTFd).

Please use the script provided to convert your code source to base64 (base64_code.py). The "EOF" is required.

Arenas

There are 3 arenas

1. **Easy PVE**: You will fight agains an easy bot from the HeroCTF.
2. **Medium PVE**: You will fight agains a medium bot from the HeroCTF.
3. **PVP**: You will fight agains the current arena champion. If you beat him, your bot will take it's place inside the arena. Current champions cannot submit new code until they are thrown off their throne.

Submissions will stop 30min before the CTF's end.

PVP Scoring

The first player to win the *Medium PVE* arena will be the first PVP champion. Other players need to beat him to take his place.
Every 10 minutes **P** points will be awarded to the current champion.
P will start at 1, when the first champion enters the PVP arena. Every half hour its value will double.
10min before the CTF's end, scoring will stop, and the team with the most points will be awarded a flag worth 10 points. It might not be much, but it could mean the different between a second or first place ;)

Game rules

Goal

In this game, two ant colonies are battling for sugar. The goal is to end up with a larger sugar reserve than the opponent. The game ends when all the sugar has been collected, if the maximum number of rounds has been reached or if any player makes an illegal move.

Grid

The grid is a square of random size, between 8x8 and 24x24 included. Your base is composed of all squares on your border of the square (column at index 0).

Your turn

Each round, the players are both provided with:

- A list of positions of their own ants
- A list of positions of their own ants one round ago
- A list of positions of the opponent's ants
- A list of positions of the opponent's ants one round ago
- A list of positions of discovered sugar cubes and the remaining suger on them
- The total amount of sugar to begin with
- The grid size
- The current score of both players
- The price of a new ant
- Some data that can be used to store information between rounds

With this information, the players must decide for each ant if they want to:

- move it
- stay and grab sugar (you can only grab sugar if you are on a sugar cube)
- stay and deposit sugar (you can only deposit sugar at the base)
- stay and do nothing

Legal moves are up, down, left, right and only one cell steps are allowed. Diagonal steps are not allowed. Players can also buy a new ant by adding an ant on an unoccupied square of the base (conditions: pos is on the base, carrying = False, move = "stay").
In the code you provide for your bot, no libraries can be imported, only vanilla Python is allowed.

Combat

An ant can't step over another ant. If two enemy ants walk on the same cell at the same time, one of them will kill the other, with a 50% chance. This scenario can't happen with friendly ants, since it would be considered an illegal move.

Sugar Cubes

To collect sugar, you need to stay one round on a sugar cube and switch the carrying variable from False to True. To deposit sugar, you need to stay one round on the base and switch the carrying variable from True to False. Switching the carrying variable in any other way will result in a illegal move. Trying to pickup sugar from an empty cube will also result in an illegal move.
Sugar cubes are discovered if an ant steps on an adjacent cell. All sugar cubes have the same amount of sugar to start with. This amount can change from one game to another.

Illegal moves

Any illegal move leads to immediate defeat. If both players make an illegal move at the same time, it's a draw. Illegal moves are :

- Moves in a unauthorized direction
- Moving on the ennemy base
- Changing the carrying variable in an unauthorized way
- Moving two friendly ants on the same cell

Data formats

Function structure

Your function needs to have a specific structure in order for the game engine to be able to use it. A template is available on the different code submission pages. The function needs to be called "make_move", take one parameter (dictionary as described below in the "Input" section) and return on variable (dictionary as described below in the "Output" section).

Input

This is the data that will be passed to your function.

Example

```
{
    "your_ants": [
        {
            "pos": (1, 1),
            "last_pos": (1, 0),
            "carrying": False,
            "last_carrying": False

        },
        {
            "pos": (1, 0),
            "last_pos": (1, 0),
            "carrying": True
            "last_carrying": False
        }
    ],
    "opponent_ants": [
        {
            "pos": (5, 3),
            "last_pos": (5, 3),
            "carrying": False
            "last_carrying": False

        }
    ],
    "discovered_cubes": [
        {
            "pos": (1, 0),
            "sugar": 5
        }
    ],
    "total_sugar_available": 68,
    "grid_size": 16,
    "your_score": 7,
    "opponent_score": 9,
    "ant_cost": 2,
    "player_data": b"AAAAAAAAAAAA"
}
```


- *your_ants*:
    - **list** containing the information of your ants
    - *your_ants.pos*:
        - **tuple of two integers** (x, y) representing the position of the ant on the grid
    - *your_ants.last_pos*:
        - **tuple of two integers** (x, y) representing the position of the ant on the grid one round ago
    - *your_ants.carrying*:
        - **boolean**, True if the ant is carrying sugar, False otherwise
    - *your_ants.last_carrying*:
        - **boolean**, True if the ant was carrying sugar one round ago, False otherwise
- *opponent_ants*:
    - **list** containing the information of your ants
    - *opponent_ants.pos*:
        - **tuple of two integers** (x, y) representing the position of the ant on the grid
    - *opponent_ants.last_pos*:
        - **tuple of two integers** (x, y) representing the position of the ant on the grid one round ago
    - *opponent_ants.carrying*:
        - **boolean**, True if the ant is carrying sugar, False otherwise
    - *opponent_ants.last_carrying*:
        - **boolean**, True if the ant was carrying sugar one round ago, False otherwise
- *discovered_cubes*
    - **list** containing the positions of the discovered sugar cubes
    - *discovered_cubes.pos*:
        - **tuple of two integers** (x, y) representing the position of the sugar cube on the grid
    - *discovered_cubes.sugar*
        - **integer** representing the amount of sugar available on the sugar cube
- *total_sugar_available*
    - **integer** representing the total amount of sugar available on the board
- *grid_size*
    - **integer** representing the size of the grid

- *your_score*
    - **integer** representing the amount of sugar you have brought back to your base
- *opponent_score*
    - **integer** representing the amount of sugar the opponent has brought back to their base
- *ant_cost*
    - **integer** representing the amount of sugar needed to create a new ant
- *player_data*
    - **bytes** representing the data of the player. This data is not used in the game, but can be used to store information between rounds.

Output

This is the data that your function needs to return.
Warning: The ant positions need to be sent back in the exact same order as they were received.

Example

```
{
    "your_ants": [
        {
            "pos": (1, 1),
            "carrying": False,
            "move": "down"

        },
        {
            "pos": (1, 0),
            "carrying": True,
            "move": "left"
        }
    ],
    "player_data": b"AAAAAAAAAAAA"
}
```

- *your_ants:*
    - **list** containing the information of your ants
    - *your_ants.pos*:
        - **tuple of two integers** (x, y) representing the position of the ant on the grid before moving
    - *your_ants.pos*:
        - **boolean**, True if the ant is carrying sugar, False otherwise
        - this can only change state from False to True on a sugar cube and from True to False on the base. Otherwise, it will result in a defeat.
        - trying to start carrying on an empty cube will result in a defeat.
    - *your_ants.move*:
        - **string** representing the move of the ant. It can be "up", "down", "left", "right" or "stay"
- *your_ants:*
    - **bytes** representing the data of the player. This data is not used in the game, but can be used to store information between rounds.
    - This variable has to be 16 bytes at most.