

paper.io.sessdsa

# AI 函数编写手册

2018 数算大作业技术组

---

## 目录

0. 代码编写规范 .....	2
1. play 函数 .....	2
接收参数 .....	2
函数返回值 .....	2
2. load 函数（可选） .....	3
接收参数 .....	3
3. summary 函数（可选） .....	3
接收参数 .....	3
4. init 函数与 summaryall 函数（可选） .....	3
接收参数 .....	3
5. 附录 .....	4
5.1. 坐标系统 .....	4
5.2. 自定义数据格式 .....	4
a. 游戏数据 stat : .....	4
d. 对局结果 match_result .....	6
e. 函数存储 storage : .....	6
5.3. 游戏执行逻辑 .....	6

---

# 0. 代码编写规范

- a. 代码文件应以.py 结尾
- b. AI 代码文件必须包含 play 函数，可以选择是否包含其余模板中所包含的函数；除此之外，不应包含与 play 函数同级的其它对象
- c. 禁止在 play 函数中使用 global 关键字或类似功能的代码，比赛命名空间的整洁需要你共同维护
- d. 禁止 import match\_core
- e. 不建议使用第三方库编写 AI，有可能被比赛环境直接视为 ImportError 而判负（若需使用特定第三方库请提前咨询技术组）
- f. 建议格式规范、注释丰富地编写代码，这样会受到三周后写报告的自己的感谢
- g. 对规则的最终解释权归技术组所有，若代码中发现违例行为，该版本代码将被取消参赛资格

## 1. play 函数

编写的 AI 函数对外的接口应命名为 play，比赛程序将按 `action = play(stat, storage)` 形式调用。

比赛系统代码会在每一回合开始时调用执行 play 函数，决定函数控制的玩家方向的改变（左转、右转、不改变方向），随后朝改变后的方向前进一单位的距离。

### 接收参数

接收参数包含两部分：游戏数据 stat 与函数存储 storage

### 函数返回值

比赛核心逻辑中使用如下逻辑执行函数返回值操作（伪代码）：

```
action = play(stat, storage)
if isinstance(action, str) and len(action) > 0:
    op = action[0].upper()
    if op == 'L': turn_left()
    elif op == 'R': turn_right()
```

若返回值为非空字符串且首字母大写为 L 或 R，则对应左转与右转操作；否则视为直行分类：

识别为“左转”：'l', 'L', 'Left', 'LEFT', 'Legendary' ...  
识别为“右转”：'r', 'R', 'right', 'RIGHT', 'Robust' ...  
识别为“前进”：None, 'iwannawin', [1, 2, 3] ...

## 2. load 函数（可选）

可选的 load 函数在比赛开始前接收初始游戏信息 stat 与存储对象 storage，可在此进行 AI 函数所需变量的声明与函数的引用等

若该函数未声明将跳过

加载数据用时将计入玩家总时间

### 接收参数

接收参数有两个：游戏数据 stat 与函数存储 storage

## 3. summary 函数（可选）

对局总结函数，每场比赛结束后运行一次

比赛运行方保证对双方 storage 字典合乎逻辑的维护

### 接收参数

接收参数包含三部分：对局结果 match\_result，游戏数据 stat 与函数存储 storage

## 4. init 函数与 summaryall 函数（可选）

分别在多局比赛开始前与结束后运行一次

init 函数可用于初始化存储空间

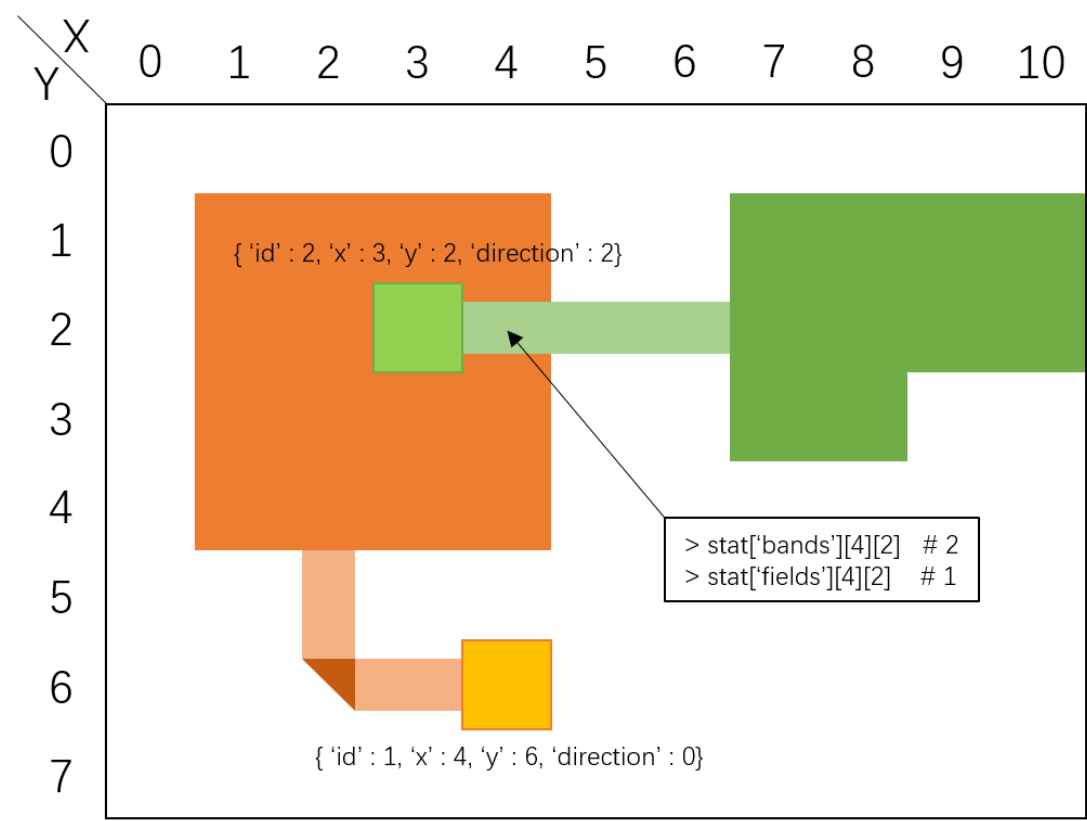
summaryall 函数可用于对一场多局比赛进行整体总结

### 接收参数

函数存储 storage

# 5. 附录

## 5.1. 坐标系



游戏数据中返回的横纵坐标均为整数，表示游戏地图的网格位置：其中 x 坐标范围为[0, 场地宽)，y 坐标范围为[0, 场地高)。fields 与 bands 二维列表中内容按列存储；调用时第一个下标为横坐标，第二个下标为纵坐标。

本文中以东南西北作为绝对方向的指代，其中 x 坐标增加的方向为东，y 坐标增加的方向为南。

## 5.2. 自定义数据格式

### a. 游戏数据 stat :

字典，包含一局游戏开始以来的相关信息  
关键字内容：

**size**：列表，包含游戏场景宽高  
类型：tuple[int]  
内容：[场地宽, 场地高]

**log**：运行记录  
类型：tuple[dict]  
内容：对局开始以来每一回合的场地信息  
**now**：当前场地信息 (=log[-1])

## b. 游戏数据-场地信息

字典，包含当前游戏状态信息

关键字内容：

**turnleft**：剩余回合数，按先后手排序  
类型：tuple[int]  
内容：[先手玩家剩余回合数, 后手玩家剩余回合数]  
**timeleft**：双方剩余思考时间（秒），按先后手排序  
类型：tuple[float]  
内容：[先手玩家剩余时间, 后手玩家剩余时间]

**fields**：纸片场地二维列表

类型：tuple[tuple[int]]

内容：fields[x][y]返回坐标(x, y)点纸片领地归属，1 代表先手玩家，2 代表后手玩家，None 代表无纸片覆盖。坐标含义详见“附：坐标系统”部分

**bands**：纸带场地二维列表

类型：tuple[tuple[int]]

内容：bands[x][y]返回坐标(x, y)点纸带领地归属，1 代表先手玩家，2 代表后手玩家，None 代表无纸带覆盖。坐标含义详见“附：坐标系统”部分

**players**：玩家信息列表，包含双方玩家信息，按先后手排序

类型：tuple[dict]

内容：[先手玩家信息, 后手玩家信息]

**me**：自己控制的玩家信息

**enemy**：对手玩家信息

以上部分具体内容详见“游戏数据-玩家信息”部分

## c. 游戏数据-玩家信息：

当前游戏状态中一个玩家的状态

内容：

**id**：玩家标记

类型：int

内容：

1：先手玩家

2：后手玩家

**x, y**：横、纵坐标

详见“坐标系统”部分

**direction**：数字标记的当前方向

类型：int

内容：

- 0：向东
- 1：向南
- 2：向西
- 3：向北

#### d. 对局结果 match\_result

长度为 2 的元组，记录了本次对局的结果

内容：

**match\_result[0]**：胜者编号（数组中下标）

- 0：先手玩家胜
- 1：后手玩家胜
- None：平局

**match\_result[1]** - 胜负原因编号

- 0：撞墙
- 1：纸带碰撞
- 2：侧碰
- 3：正碰，结算得分
- 4：领地内互相碰撞
- 1：AI 函数报错
- 2：超时
- 3：回合数耗尽，结算得分

#### e. 函数存储 storage：

字典，可供玩家自由使用，存储数据；在对局之间不会自动清空。

### 5.3. 游戏执行逻辑

