主办方提供的SDK接口

特别注意：玩家程序的主入口是player.py文件，这个文件不可以更改名称，并且玩家最终提交的代码必须包含这个文件！！！

为提高代码可读性，现提出如下命名规定

命名规定：

- A方为参赛队伍，B方为敌方（内置AI程序）
- A方玩家各智能体名称为：`uavA1`，`uavA2`，`uavA3`，`carA1`，`carA2`，`carA3`
- B方玩家各智能体名称为：`uavB1`，`uavB2`，`uavB3`，`carB1`，`carB2`，`carB3`

以下接口使用实例以A方(我方)1号无人机`uavA1`，与B方（敌方）2号无人车`carB2`，B方基地`GCSB0`为例，（事实上，敌方的名字是需要通过以下提供的有关接口去获取的，为便于阅读，假设已知敌方基地与编号为2的无人车的名字）

提供的接口

1.设置移动指令(x，y，z方向速度，偏航角)

Function：`self.set_move_cmd(name, cmd)`

Parameters： cmd为长度为4的列表

cmd[0] : x方向的速度 (linear_x)

cmd[1] : y方向的速度 (linear_y)

cmd[2] : z方向的速度 (linear_z)

cmd[3] : 偏航角速度 (angular_z)

以上4个括号内的名字仅供命名参考

e.g.

```
linear_x=1

linear_y=0

linear_z=0

angular_z=0

cmd=[linear_x,linear_y,linear_z,angular_z]

self.set_move_cmd('uavA1', cmd)
```

效果： 名为`uavA1`的无人机，其移动指令`cmd`被设置为（并不会真正的移动）

沿x轴正方向移动1单位长度/单位时间 (真正的执行请看接口3)

2.获取位置信息与姿态信息

Function：`self.get_pose(name)`

Parameters&Return value：长度为6的列表，

从左至右依次为x，y，z（位置，前3位），roll，pitch，yaw（姿态，后3位）分别为滚转角，俯仰角，偏航角

e.g. `carb2_position = self.get_pose('carB2')[0:3]`

效果：获取无人车B2的位置信息（x, y, z）存到变量carb2_position中

[0:3]表示起始索引为0，长度为3，用来获取carB2位置信息，不要其姿态信息

3.执行移动指令：执行设置后所有智能体的移动指令，如果不移动则请将移动指令参数置为0，否则会默认执行上一次的移动指令

Function：`self.move_cmd_send()`

e.g.

```
self.set_move_cmd('uavA1', cmd)

self.set_move_cmd('uavA2', cmd)

self.set_move_cmd('uavA3', cmd)

self.set_move_cmd('carA1', cmd)

self.set_move_cmd('carA2', cmd)

self.set_move_cmd('carA3', cmd)

self.move_cmd_send()
```

效果：对这6个智能体执行接口1的示例中设置好的移动指令

（可以使用循环简化代码，但要注意car没有y，z方向的速度）

4.发送攻击指令（名字为name的智能体向target射出子弹，target为目标位置，即长度为3的列表，元素分别为目标点x，y，z坐标）

Function：`self.attack_cmd_send(name, target)`

e.g.

```
target1 = self.get_pose('carB2')[0:3]   # get the position of carB1 of enemy
team

self.attack_cmd_send('uavA1', target1)  # ‘uavA1’send the attack command
```

即uavA1向carB2所处的位置发出子弹

5.获取血量的信息（可获取的目标：A方，B方智能体及各自基地）

Function：`self.get_blood(name)`

Parameters&Return value: (int)

6.获取弹药量的信息

Function：`self.get_remain_bullet(name)`

Parameters&Return value：(int)

7.判断是否存活

Function：`self.is_alive(name)`

Return value： (bool)

8.获取我方基地的名字

Function：`self.get_self_basestation_name()`

Return value：name(string)

9.获取敌方基地的名字

Function：`self.get_enemy_basestation_name()`

Return value：name(string)

10.获取我方所有智能体的名字(包括无人机与车)

Function：`self.get_self_agent_name_list()`

Return value： [string] 字符串列表，元素为我方所有无人机与车的名字

11.获取敌方所有智能体的名字(包括无人机与车)

Function：`self.get_enemy_agent_name_list()`

Return value： [string] 字符串列表，元素为敌方所有无人机与车的名字

参赛者需要编写的内容

1、直接使用提供的例子并在其基础上修改

2、根据SDK提供的接口自行编写代码逻辑

如何导入接口的包（在player文件夹内部）：

`from player_interface import PlayerInterface`

如何继承主办方提供的接口以及初始化：

```
class 类的名字(PlayerInterface):
    def __init__(self):
        super(类的名字, self).__init__()

e.g.
class Example(PlayerInterface):
    def __init__(self):     #### initialization
        super(Example, self).__init__()
```

为保障实际应用的安全性，建议初始化的同时设置最大速度（SDK内部也有速度上限保护，此处参数仅用于个人调试）

e.g. `self.v_max=0.5`