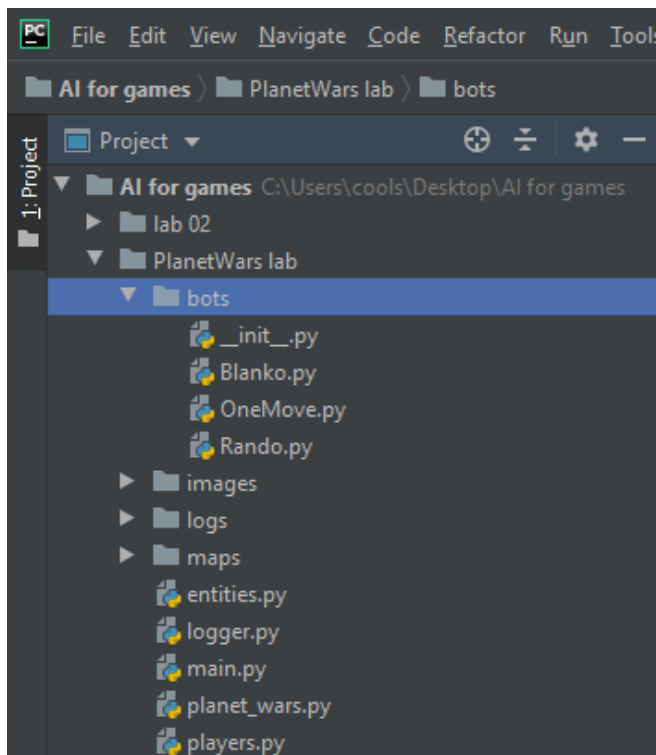


Created a new Bot called it Rando inside the bot directory



Then added some code that was already provided for the new bot and used it, while also amending the `players = [\"\"]` called in the main file, so that it launches one fleet at a time.

```
if gameinfo.my_fleets:
    return
```

```
if __name__ == '__main__':
    gamestate = open('./maps/map5.txt').read()
    players = ['Blanko', 'OneMove', 'Rando']
    window = PlanetWarsWindow(gamestate=gamestate, players=players, max_game_length=500)
    app.run()
    window.game.logger.flush()
```

Then

Used the code provided to carry out below functionality

Time to attack! The basic design is this:

1. Make sure we have a planet we can launch a fleet from. (Do you know why is this?)
2. Make sure there is at least one planet to attack! (Again, make sure you know why this is.)
3. Select a random source planet from the planets we have (`my_planets`).
4. Select a random target planet from the planets that are not in our control (`not_my_planet`)
5. Launch a fleet if the sources planet has more than 10 ships.
6. Only launch 75% of the ships available from the source planet.

Here's one way to do all that:

```
# check if we should attack
if gameinfo.my_planets and gameinfo.not_my_planets:
    # select random target and destination
    dest = choice(list(gameinfo.not_my_planets.values()))
    src = choice(list(gameinfo.my_planets.values()))
    # launch new fleet if there's enough ships
    if src.num_ships > 10:
        gameinfo.planet_order(src, dest, int(src.num_ships * 0.75) )
```

To change the behaviour of the bot from shooting random targets we used the code provided below: where it used the destination point by finding a planet with the minimum number of ships.

```
src = max(gameinfo.my_planets.values(), key=lambda p: p.num_ships)
dest = min(gameinfo.not_my_planets.values(), key=lambda p: p.num_ships)
```

Rando bot:

```
Rando.py x Blanko.py x main.py x
1 from random import choice
2
3
4 class Rando(object):
5
6     def update(self, gameinfo):
7
8         # only send one fleet at a time
9
10        # check if we should attack
11        if gameinfo.my_planets and gameinfo.not_my_planets:
12            if gameinfo.my_fleets:
13                return
14            # select random target and destination
15
16            src = max(gameinfo.my_planets.values(), key=lambda p: p.num_ships)
17            # Find a target planet with the minimum number of ships.
18            dest = min(gameinfo.not_my_planets.values(), key=lambda p: p.num_ships)
19
20            # launch new fleet if there's enough ships
21            if src.num_ships > 10:
22                gameinfo.planet_order(src, dest, int(src.num_ships * 0.75))
23
```

Tried using the other provided bot blanko and tested the inverse proportional search

```
class Blanko(object):
    def update(self, gameinfo):
        # check if we should attack
        if gameinfo.my_planets and gameinfo.not_my_planets:
            if gameinfo.my_fleets:
                return
            # select random target and destination

            src = max(gameinfo.my_planets.values(), key=lambda p: p.num_ships)
            # using an inverse proportional maximum search to determine the planets with max num of ships
            dest = max(gameinfo.not_my_planets.values(), key=lambda p: 1.0 / (1 + p.num_ships))
            # launch new fleet if there's enough ships
            if src.num_ships > 10:
                gameinfo.planet_order(src, dest, int(src.num_ships * 0.75))

Blanko > update() > if gameinfo.my_planets and game...

main x
C:\Users\cools\AppData\Local\Programs\Python\Python38-32\python.exe "C:/Users/cools/Desktop/AI for games/Plan
Process finished with exit code 0
```

The output was something like this:

