

Answer for Lab1 Go

File structure

```
|—— test file
    |—— test_0.txt
    |—— test_1.txt
    |—— test_2.txt
    |—— test_3.txt
    |—— test_4.txt
    |—— test_5.txt
    |—— test_6.txt
|—— Code file
    |—— chess_demo.py // this code is provide by textbook
    |—— chess_go.py  // this code is done by myself
|—— README
```

Trick: how to check your answer with standard solution efficient?

problem 1: problem 1 only has True and False, so you can print it. It will be very easy.

problem 2: for problem 2, each test file only has few points, so you can print them and check with standard answer.

problem 3: for problem 3, the solution contain many points, if you check it in the terminal or file one by one, it'll be very tired for your brain and eyes. So I suggest you a method below.

Add a function like this: input parameter is a list of all impossible position white chess can be placed.

e.g. `[[0,1],[0,2],[0,3]]`

```
def plot_3(ans):
    a = np.zeros((BOARD_SIZE, BOARD_SIZE))
    for point in ans:
        row = point[0]
        col = point[1]
        a[row, col] = COLOR_BLACK
    plot_go(a, 'answer')
```

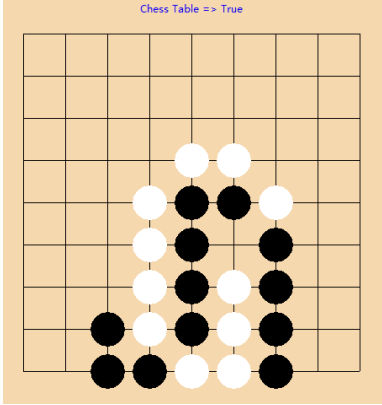
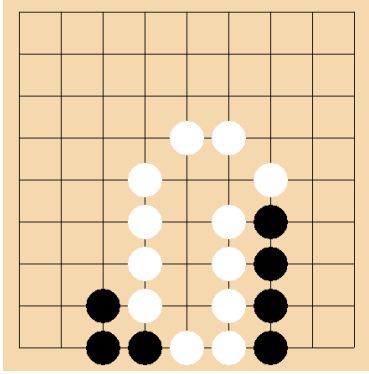
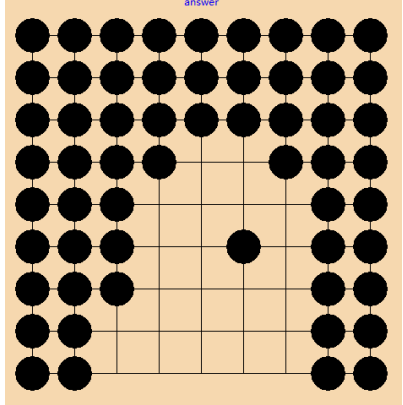
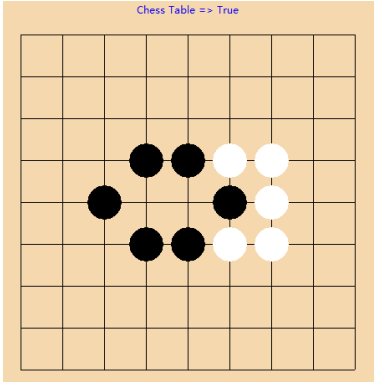
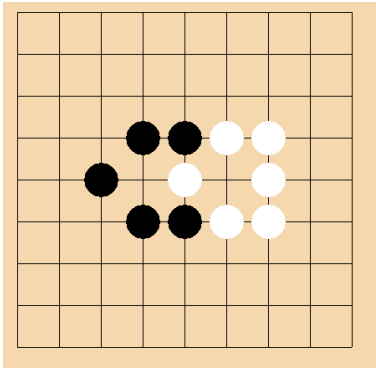
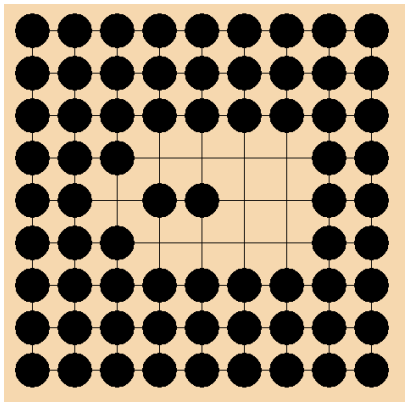
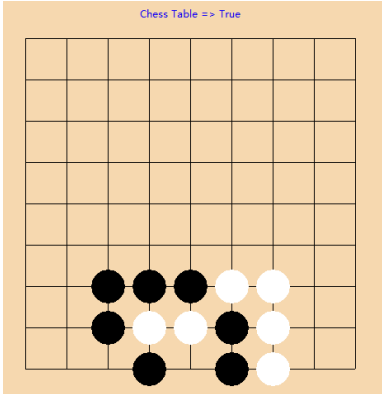
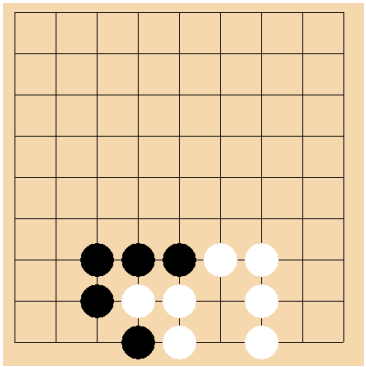
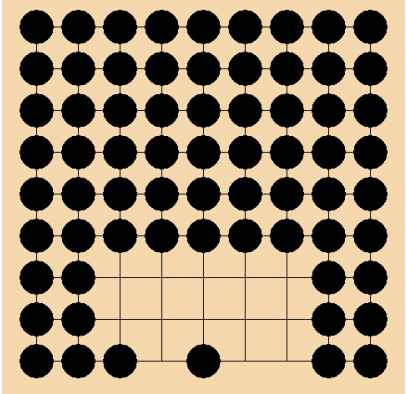
Code

Two version code:

1. standard code provide by textbook
2. code by myself.

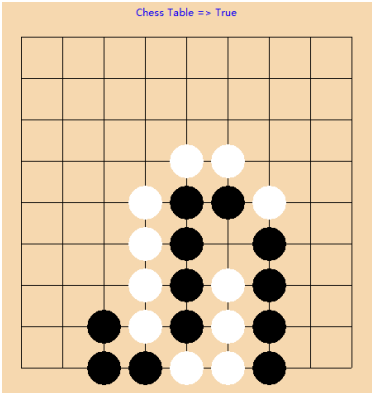
Answer

PS: answer I give below is based on my trick.

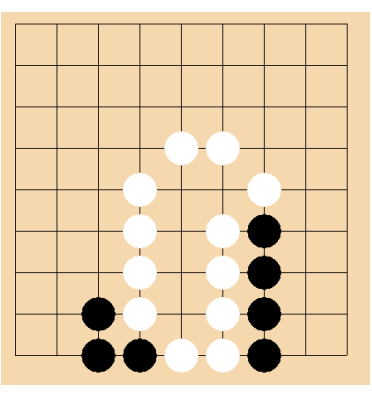
test_0.txt		
1. True	2. [5, 5]	3. ...
		
test_1.txt		
1. True	2. [4, 4]	3. ...
		
test_2.txt		
1. True	2. [8, 4]	3. ...
		

test_3.txt

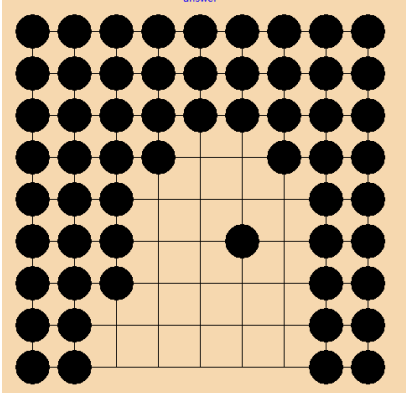
1. True



2. [5, 5]

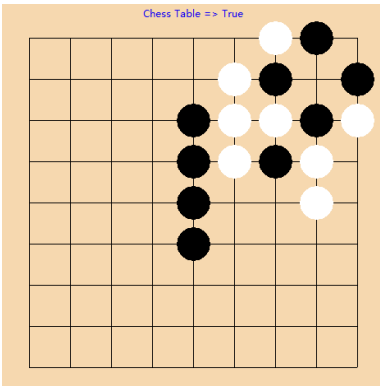


3. ...

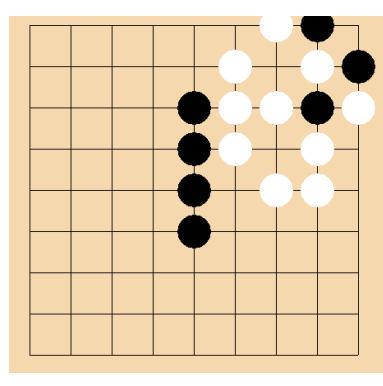


test_4.txt

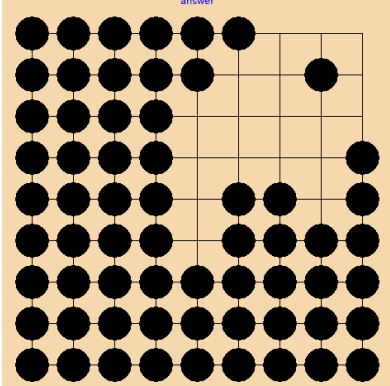
1. True



2. [4, 6], [1, 7]

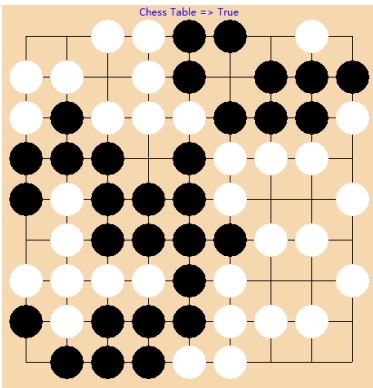


3. ...

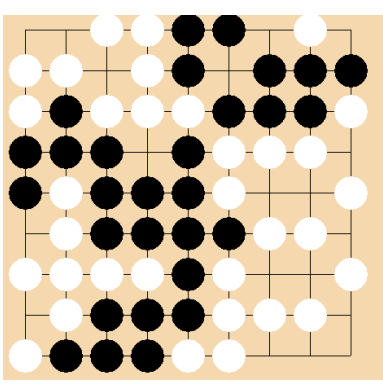


test_5.txt

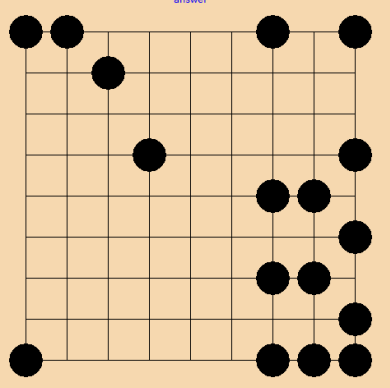
1. True



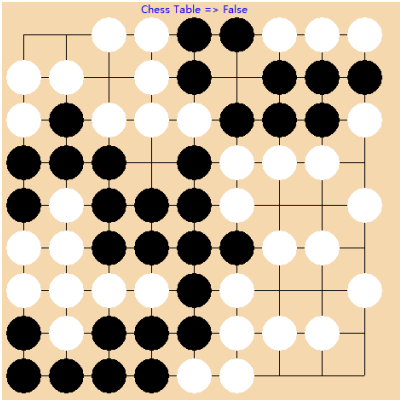
2. [8, 0]



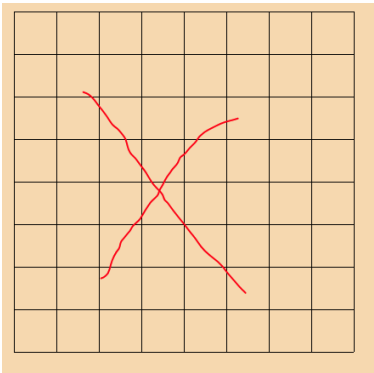
3. ...



1. False



2. []



3. ...

