Lecture Notes:Week 6

From JOCWiki

Contents

[hide]

TIC TAC TOE

[edit]

Hello All, So in this modue we will be fcusing on a simple game called tic-tac toe, yes the same game many of us played in young days or even now, basically on notebook back:), but in a way such that its fun , bascially a programed version of it and most important no way to cheat!!! So let get a intro about it...

Intro

[edit]

Tic-tac-toe(genral as we call) , noughts and crosses(Americans) or Xs and Os(British) all surround to a simmple game having attributes as:
It is a paper-and-pencil game for two players, X and O,
Player takes turns marking the spaces in a 3×3 grid.
The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row wins the game.

Quite simple isn't it, but if u infer u can see its not so simple , for ones who don,t know the optimal strategy its not a simple one and for us as coders it is used to teach **AI** *and sportsmanship.*
As Said we need to have an optimal strategy so whats the way lets try to find:)

Optimal Strategy
[edit]

Win: If the player has two in a row, they can place a third to get three in a row.
Block: If the opponent has two in a row, the player must play the third themselves to block the opponent.
Fork: Create an opportunity where the player has two ways to win (two non-blocked lines of 2).
Blocking an opponent's fork: If there is only one possible fork for the opponent, the player should block it. Otherwise, the player should block all forks in any way that simultaneously allows them to create two in a row. Otherwise, the player should create a two in a row to force the opponent into defending, as long as it doesn't result in them creating a fork. For example, if "X" has two opposite corners and "O" has the center, "O" must not play a corner in order to win. (Playing a corner in this scenario creates a fork for "X" to win.)
Center: A player marks the center. (If it is the first move of the game, playing on a corner gives the second player more opportunities to make a mistake and may therefore be the better choice; however, it makes no difference between perfect players.) Opposite corner: If the opponent is in the corner, the player plays the opposite corner.
Empty corner: The player plays in a corner square.
Empty side: The player plays in a middle square on any of the 4 sides.

Optimal Strategy Can Be inferred from this:

```
Play your first X in a corner. ...


Try to win if your opponent plays the first O in the center. ...


Win automatically if your opponent plays his first O in any square besides the center. ...


Place your third X so you have two possible winning moves. ...


Win with your fourth X.
```

So now we have know the optimal strategy, lets try to work on a program which can be cheated:)

Prerequisites
[edit]

Before we further go on to code we need some understanding of certain things before hand , so now let us get know to it

```
)
```

Matrix
[edit]

*As we know we need to use a 3*3 grid , we use a matrix ,same concept we study in maths but here is a twist as we know matix start from 1,1 and goes upto 3,3 , but due to indexing we start from 0,0 such that:*

```
1 // 3 x 3 matrix<br>
2
3       0,0  0,1  0,2
4 M =   1,0  1,1  1,2
5       2,0  2,1  2,2
```

*So this is what we call matrix , and it is a powerful concept in game development along with matrix programs and to code we need to use two loop one for rows and one for cols and increment cols one by one***(cols+1)**...
We will see more of it in coding part:)

Numpy Library(Arrays)
[edit]

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python.
Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arrays in Numpy

For more info on Numpy refer here:Numpy

Working Of Numpy
[edit]

Can be imported using

```
1  '''import numpy # imports Numpy Library
```

Array in Numpy is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In Numpy, number of dimensions of the array is called **rank** *of the array. A tuple of integers giving the size of the array along each dimension is known as shape of the array. An array class in Numpy is called as ndarray. Elements in Numpy arrays are accessed by using square brackets and can be initialized by using nested Python Lists.*

Functions
[edit]

I know its already been know to all but yet for revision we are discussing it here

So functions are' set of specific instruction used to perform specific tasks. We have used many of inbuilt function of python such as **import**,**delete**, **input()**, **split()** *and much more. But many programming languages provide us the flexibility/ ability t write our own function and call it , it has two main syntax namely*

```
def function name(parameters): # creating function<br>
# body of function<br>
function name(parameters)# Calling function<br>
```

So Now we have know all the requirements we can now use our knowledge to write a program which is imitating tic tac toe

PROGRAM FOR TIC TAC TOE:[edit]

```
import numpy
board=numpy.array([['-','-','-'],['-','-','-'],['-','-','-']])
p1s='X'
p2s='O'

def check_rows(symbol):
    for r in range(3):
        count=0
        for c in range(3):
            if board[r][c]==symbol:
                count =count+1
```

```
        if count==3:
            print(symbol,'won')
            return True
    return False
def check_cols(symbol):
    for c in range(3):
        count=0
        for r in range(3):
            if board[r][c]==symbol:
                count =count+1
        if count==3:
            print(symbol,'won')
            return True
    return False
def check_diagonals(symbol):
    if board[0][2]==board[1][1] and board[1][1]==board[2][0]
        print(symbol,'won')
    if board[0][0]==board[1][1] and board[1][1]==board[2][2]
        print(symbol,'won')
        return True
    return False
def won(symbol):
    return check_rows(symbol) or check_cols(symbol) or check_diagonals(symbol)
def place(symbol):
    print(numpy.matrix(board))
    while(1):
        row=int(input('Enter row -1 or 2 or 3:'))
```

```
        col=int(input('Enter column -1 or 2 or 3:'))
        if row>0 and row<4 and col>0 and col<4 and board[row-1][col-1]=='-':
            break
        else:
            print('Invalid input,please try again with valid one')
    board[row-1][col-1] = symbol
def play():
    for turn in range(9):
        if turn%2==0:
            print('X turn')
            place(p1s)
            if won(p1s):
                break
        else:
            print('O turn')
            place(p2s)
            if won(p2s):
                break
    if not(won(p1s)) and not(won(p2s)):
        print('Draw')
play()
```

OUTPUT:

[edit]

```
X turn
```

```
[['-','-','-']
 ['-','-','-']
 ['-','-','-']]

Enter row -1 or 2 or 3: 2
Enter column -1 or 2 or 3: 2

0 turn
[['-','-','-']
 ['-','X','-']
 ['-','-','-']]

Enter row -1 or 2 or 3: 1
Enter column -1 or 2 or 3: 2

X turn
[['-','O','-']
 ['-','X','-']
 ['-','-','-']]

Enter row -1 or 2 or 3: 2
Enter column -1 or 2 or 3: 1

0  turn
[['-','O','-']
 ['X','X','-']
 ['-','-','-']]
```

```
29
30 Enter row -1 or 2 or 3:1
31 Enter column -1 or 2 or 3:1
32
33 X turn
34 [['O','O','-']
35  ['X','X','-']
36  ['-','-','-']]
37
38 Enter row -1 or 2 or 3: 2
39 Enter column -1 or 2 or 3: 3
40 X won
41 X won
```

Happy Learning Binaries ...