**CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Serial no.** | **Title** | **Page no.** |
| 1. | Introduction |  |
| 2. | Acknowledgement |  |
| 3. | System requirements |  |
| 4. | Basic Algorithm |  |
| 5. | Modules |  |
| 6. | User Defined Functions |  |
| 7. | Variables |  |
| 8. | Source Code |  |
| 9. | Screen shots |  |
| 10. | Bibliography |  |

INTRODUCTION

The program is

ACKNOWLEDGMENT

SYSTEM REQUIREMENTS

* System requirements:

1. 1 gigahertz (**GHz**) or faster 32-bit (**x86**) or 64-bit (**x64**) processor
2. Keyboard
3. Colour Monitor
4. Speakers

* Software requirements:

1. Windows 7 or higher
2. Python 2.7.4 or higher
3. Pygame for Python 2.7.4

BASIC ALGORITHM

**MODULES AND LIBRARY FUNCTIONS**

|  |  |  |
| --- | --- | --- |
| **Module** | **Library Functions** | **Usage** |
| os | .remove | To remove an existing text file |
| .rename | To rename the text files |
| .path.isfile | To check if a file exists in the specified location |
| random | .randint | Generating random integer |
| .shuffle | Shuffling the elements of a list |

**User Defined Functions**

|  |  |
| --- | --- |
| **Function name** | **Task** |
|  |  |
|  |  |
|  |  |

**Variables**

|  |  |
| --- | --- |
| **Data Type** | **Variable name** |
| String |  |
| List |  |
| Integer/Floating Point |  |
| Class objects |  |
| Boolean |  |

**SOURCE CODE**

import random , os

corner=[(0,0),(2,2),(0,2),(2,0)]

edge=[(0,1),(2,1),(1,0),(1,2)]

center = [(1,1)]

random.shuffle(corner)

random.shuffle(edge)

diagonal1 = [(i,j) for i in range(3) for j in range(3) if i == j]

diagonal2 = [(i,j) for i in range(3) for j in range(3) if (i+j) == 2]

priority = center + corner + edge

class Errors(Exception):

pass

OccupiedError=Errors("Enter a location which is free")

class basic:

def \_\_init\_\_(self):

self.grid=[[0 for i in range(3)]for j in range(3)]

def display(self):

print " " + "-"\*11

for i in range(3):

print "|",

for j in range(3):

if self.grid[i][j]==0:

print " " , "|" ,

else :

print self.grid[i][j], "|" ,

print

print " " + "-"\*11

def chkinp(self): ##Check if input is valid

while True:

try:

err1=True

row , waste , col = tuple(raw\_input())

err1=False

row,col=int(row),int(col)

if row not in (1,2,3) or col not in (1,2,3):

raise IndexError

if self.grid[row-1][col-1]<>0:

raise OccupiedError

self.grid[row-1][col-1]="X"

break

except ValueError , e:

if err1==False:

print "Enter a number for row or column value"

else:

print "Input in the given format"

print "row<space>column"

except IndexError:

print "Enter a valid location"

except Errors as e:

print e

def win(self,key): ##check for winning combo

if self.grid[1][1]==self.grid[2][2]==self.grid[0][0] and self.grid[1][1]<>0: #Principal Diagonal

print key, "WON"

return True

elif self.grid[0][2]==self.grid[1][1]==self.grid[2][0] and self.grid[1][1]<>0: #right Diagonal

print key, "WON"

return True

elif self.grid[0][1]==self.grid[0][2]==self.grid[0][0] and self.grid[0][1]<>0: #Row 1

print key, "WON"

return True

elif self.grid[1][1]==self.grid[1][2]==self.grid[1][0] and self.grid[1][1]<>0: #Row 2

print key, "WON"

return True

elif self.grid[2][1]==self.grid[2][2]==self.grid[2][0] and self.grid[2][1]<>0: #Row 3

print key, "WON"

return True

elif self.grid[1][0]==self.grid[2][0]==self.grid[0][0] and self.grid[1][0]<>0: #Col 1

print key, "WON"

return True

elif self.grid[1][1]==self.grid[2][1]==self.grid[0][1] and self.grid[1][1]<>0: #Col 2

print key, "WON"

return True

elif self.grid[0][2]==self.grid[1][2]==self.grid[2][2] and self.grid[2][2]<>0: #Col 3

print key, "WON"

return True

class easy(basic):

def compplay(self): ##Computer random guess

row,col = self.getmove()

while self.grid[row][col]<>0:

row,col = self.getmove()

else:

self.grid[row][col]="O"

def getmove(self):

row=random.randrange(3)

col=random.randrange(3)

return row,col

def exiter(self): ##function to check if all locations are filled

s=0

for i in self.grid :

if 0 in i:

return False

else:

return True

class hard(easy):

def getmove(self):

### Comp try win

for i in range(3):

if (self.grid[i].count("O") == 2) and (not all(self.grid[i])): ##Check row

return i, self.grid[i].index(0)

column = [self.grid[j][i] for j in range(3)] ##Check column

if (column.count("O") == 2) and (not all(column)):

return column.index(0),i

diag1 = [self.grid[x][y] for x,y in diagonal1] ##Check right diagonal

if (column.count("O") == 2) and (not all(column)):

return column.index(0), column.index(0)

diag2 = [self.grid[x][y] for x,y in diagonal2] ##Check left diagonal

if (column.count("O") == 2) and (not all(column)):

return column.index(0), 2-column.index(0)

### Block player

for i in range(3):

if (self.grid[i].count("X") == 2) and (not all(self.grid[i])): ##Check row

return i, self.grid[i].index(0)

column = [self.grid[j][i] for j in range(3)] ##Check column

if (column.count("X") == 2) and (not all(column)):

return column.index(0),i

diag1 = [self.grid[x][y] for x,y in diagonal1] ##Check right diagonal

if (column.count("X") == 2) and (not all(column)):

return column.index(0), column.index(0)

diag2 = [self.grid[x][y] for x,y in diagonal2] ##Check left diagonal

if (column.count("X") == 2) and (not all(column)):

return column.index(0), 2-column.index(0)

# If the user or computer is not winning

for row,col in priority:

if not self.grid[row][col]:

return row,col

def player(game,lev): ##player plays ""01st"" input and result of game

while True:

game.display()

print "Your turn"

print "Enter row number followed by column number as"

print "row<space>column"

game.chkinp()

game.display()

if game.win('YOU'):

update\_stats('YOU',lev)

break

elif game.exiter():

update\_stats('TIE',lev)

print "It is a tie"

break

game.compplay()

print "Computer played"

if game.win('COMPUTER'):

update\_stats('COMPUTER',lev)

game.display()

break

def comp(game,lev): ##computer plays ""01st"" and result

while True:

game.compplay()

game.display()

if game.win('COMPUTER'):

update\_stats('COMPUTER',lev)

break

if game.exiter():

update\_stats('TIE',lev)

print "It is a tie"

break

print "Your turn"

print "Enter row number followed by column number as"

print "row<space>column"

game.chkinp()

if game.win('YOU'):

update\_stats('YOU',lev)

game.display()

break

def update\_stats(key,level):

f=open("./stats/" + name + ".txt" ,"r")

nf=open("new.txt", "w")

if key=='YOU':

l=f.readlines()

if level=='1':

for i in range(len(l)):

if i==2:

no=str(int(l[i][6:-1])+1)

new=l[i][:6]+no

nf.write(new+'\n')

else:

nf.write(l[i])

else:

for i in range(len(l)):

if i==6:

no=str(int(l[i][6:-1])+1)

new=l[i][:6]+no

nf.write(new+'\n')

else:

nf.write(l[i])

elif key=='COMPUTER':

l=f.readlines()

if level=='1':

for i in range(len(l)):

if i==3:

no=str(int(l[i][7:-1])+1)

new=l[i][:7]+no

nf.write(new+'\n')

else:

nf.write(l[i])

else:

for i in range(len(l)):

if i==7:

no=str(int(l[i][7:-1])+1)

new=l[i][:7]+no

nf.write(new+'\n')

else:

nf.write(l[i])

else:

l=f.readlines()

if level=='1':

for i in range(len(l)):

if i==4:

no=str(int(l[i][5:-1])+1)

new=l[i][:5]+no

nf.write(new+'\n')

else:

nf.write(l[i])

else:

for i in range(len(l)):

if i==8:

no=str(int(l[i][5:-1])+1)

new=l[i][:5]+no

nf.write(new+'\n')

else:

nf.write(l[i])

f.close()

nf.close()

os.remove("./stats/" + name + ".txt")

os.rename("new.txt","./stats/" + name + ".txt")

def writefile():

f=open("./stats/" + name + ".txt","w")

f.write(name+"'s stats \n")

f.write("LEVEL --- EASY\n")

f.write("Wins: 0\n")

f.write("Losses: 0\n")

f.write("Ties: 0\n")

f.write("LEVEL --- HARD\n")

f.write("Wins: 0\n")

f.write("Losses: 0\n")

f.write("Ties: 0\n")

f.close()

def initialise():

global name ,times

times=1

name=raw\_input("Enter your name: ")

name=name.upper()

if not(os.path.isfile("./stats/" + name + ".txt")): ##Check if file exists

print 'Hi!' , name

print 'Tic-Tac-Bot is the best Tic-Tac-Toe player the world has ever seen!!!'

print 'You have been challenged by it for a game of Tic-Tac-Toe'

print

start=raw\_input("Press 'ENTER' key to continue")

print " You are 'X' "

print

writefile()

else:

print "Welcome back!" , name

def playgame(): ##base function

global times

ch="yes"

initialise()

level=0

while level<>'6' :

level=raw\_input('''Choose an option:

1)Easy

2)Hard

3)Statistics

4)Reset Statistics

5)Change Player

6)Exit

''')

while level not in ('1','2','3','4','5','6'):

level=raw\_input("Enter a valid choice: ")

if level=='1':

print "Game" , times

if times%2==1:

ob=easy()

player(ob,level)

else:

ob=easy()

comp(ob,level)

times+=1

elif level=='2':

print "Game" , times

if times%2==1:

ob=hard()

player(ob,level)

else:

ob=hard()

comp(ob,level)

times+=1

elif level=='3':

f=open("./stats/" + name + ".txt","r")

print f.read()

f.close()

elif level=='4':

writefile()

elif level=='5':

initialise()

print "Thank You"

playgame()

**SCREENSHOTS**

**BIBLIOGRAPHY**