```python
"""
============================== TicTacPro ==============================
FILE: Acheivements.py
MODIFIED: 15/11/2015
STATUS: Complete
FILE DESCRIPTION:
The achievements.py file is used for loading and saving the achievements stats to a binary file for
a more permantant as it means the variables can be tranferred between instances of the game running
rather than being local to each instance of the program being run.
USAGE:
Used for saving and loading from files
"""

from main import *
from logic import *
import pickle

def achievereset():
    #resets the acheivements file
    f = open("achievements.pickle", "wb")
    stats = [0,0,0,0,0]
    pickle.dump(stats,f)
    f.close()

def achievements(res):
    #loads the stats from a local file
    f = open("achievements.pickle","rb")
    stats = pickle.load(f)
    f.close()
    #1-games played, 2-games won 3-games lost 4 - games draw 5 - level
    if res == "won":
        #ups the stats
        stats[0]+=1
        stats[1]+=1
        stats[4]+=1
        print("You are level " + str(stats[4]))
    elif res == "lost":
        if int(stats[4]) >1 :
            stats[0]+=1
            stats[2]+=1
            stats[4]-=0.5
            print("You are level "+ str(stats[4]))
        if int(stats[4]) <= 1:
            stats[0]+=1
            print("You are still on level 1!")
    elif res == "draw":
        stats[0]+=1
        stats[3]+=1
        stats[4]+=0.5
        print("You are level " + str(stats[4]))
    #closes the stats file
    f = open("achievements.pickle","wb")
    pickle.dump(stats, f)
    f.close()

def achievementsview():
    #clears the last event and sets up the window for the viewing the achievements
    pygame.event.clear()
    screen=pygame.display.set_mode((610, 650))
    achievementsmenu=pygame.image.load("images/menu/settings/achievementsmenu.png")
    screen.blit(achievementsmenu,(0,0))
    #loads the achievements
    f = open("achievements.pickle","rb")
    stats = pickle.load(f)
    f.close()
    #prints them out in the corresponding locations
    print("Games played ....................................... " + str(stats[0]),70,140)
    print("Games won ........................................... " + str(stats[1]),70,200)
    print("Games lost ............................................ " + str(stats[2]),70,260)
    print("Games drawn ....................................... " + str(stats[3]),70,320)
    print("You are level " + str(stats[4]),200,380)

    q=False
    while not q:
        pygame.display.flip()
        ev = pygame.event.get()
        for event in ev:
            if event.type == pygame.MOUSEBUTTONUP:
                #checks for the back button being pressed
                pos = pygame.mouse.get_pos()
                if pos[0] in range(23,204) and pos[1] in range(540,600):
                    return None

if __name__=="__main__":
    achievements("won")
```

```
###NEWFILE




"""
============================== TicTacPro ==============================
FILE: chat.py
MODIFIED: 27/11/2015
STATUS: Complete
FILE DESCRIPTION:
    The chat.py file contains all the functions relating to displaying the chat and
    input and the chat list.

USAGE:
This is a client chat display and input management function.
"""
from main import *
from logic import *

def main():
    screen=pygame.display.set_mode((900, 650))
    pygame.draw.rect(screen, (0,0,0), [610, 0, 300, 700], 0)
    chat=[]
    displaystring=""
    turn=True
    while True:
        #how i get mouse clicks this is event driven programming
        ev = pygame.event.get()
        for event in ev:
            if event.type == pygame.QUIT:
                raise SystemExit
            elif event.type == pygame.KEYDOWN:
                displaystring,chat=chatinput(event.key,displaystring,chat)

def chatinput(eventkey,displaystring,chat):
    #this function runs when a key is presed, and decides what to do with it

    if eventkey in range(97,123) or eventkey in range(48,58):
        #these ranges are a-z and 0-9, as it is the ascii value it is simple to
        #convert the number to a letter or number
        displaystring+=str(chr(eventkey))

    elif eventkey == 13:
        #when the enter key is pressed, sends the message to the server and also
        #it clears the display string and adds it to the chat list, then draws the chat
        if displaystring!="":
            chat.insert(0,("You",displaystring))
            sendmsg=displaystring
        displaystring=""
        drawlist(chat)

    elif eventkey == 32:
        #whhen the space key is pressed a blank space in inserted into the displaystring
        displaystring+=" "

    elif eventkey == 8:
        #when the backspace key is pressed the last letter in the displaystring is removed
        displaystring=displaystring[:-1]

    #display the current input message
    showinput(displaystring)
    return displaystring,chat

def showinput(string):
    string=(": "+string)
    #covers up any previous stuff on the screen
    pygame.draw.rect(screen, (0,0,0), [610, 600, 300, 700], 0)
    #draws texts on screen
    font = pygame.font.Font(None, 16)
    text = font.render(string, 4, (255, 255, 255))
    screen.blit(text, (620,610))
    #updates screen to show new changes
    pygame.display.flip()

def drawlist(chat):
    #set the bottom height of the screen
    y=540
    #deletes the oldest message in the chat after it gets to 14 messages
    if len(chat)>14: del chat[-1]
    pygame.draw.rect(screen, (0,0,0), [610, 0, 300, 700], 0)
    for each in chat:
        #generates the output string
        each=(str(each[0]) +": " + str(each[1]))
        #decides how to handle printing out the statement, if it will fit on one line or not
        if len(each)<47:
            #prints the string out at the y coord
            font = pygame.font.Font(None, 16)
            chatstuff = font.render(each, 4, (255, 255, 255))
            screen.blit(chatstuff, (620,y))
            #moves the y coord down to accomodate the next message
            y-=40

        elif len(each)>120:
            #word limit
            print("Too long to display!")

        else:
            #smaller recursive loop like the one above for multi line messages
            y2=y
            y-=40
            while len(each)!=0:
                #takes one lines worth of code from the each string
                line=each[:40]
                each=each[40:]
                font = pygame.font.Font(None, 16)
                chatstuff = font.render(line, 4, (255, 255, 255))
                screen.blit(chatstuff, (620,y2))
                #moves it down slightly lower to give a multiline effect
                y2+=10

        pygame.display.flip()
if __name__=="__main__":
    main()





###NEWFILE




"""
============================== TicTacPro ==============================
FILE: FirstGo.py
MODIFIED: 15/11/2015
STATUS: Complete
FILE DESCRIPTION:
The FileGo.py file is used for deciding who is the first person to go first
by asking a question and using this to determine who goes first
"""

from main import *
from logic import *
import random

def askquestion():
    #possible questions
    #structure
    #0 - question
    #1 - answer 1
    #2 - answer 2
    #3+4 ""
    #5 - correct answer number
    questions = [
["Who is now the American President?","Barrack Obama", "Sean Paul", "Michael Jordan", "Skepta", 1],
["How many cheeks do you have?","2", "0", "3", "4", 1],
["What goes up and never comes down?","Age", "Football", "Plane", "Rain", 1],
["What is the square root of 144 equal?","12.5", "12", "1", "2", 2],
["Who made Microsoft?","Bill Smith", "Bill Phil", "Bill Gates", "Bill Paul", 3],
["What is half of 200?","150", "120", "100", "102", 3],
["Which of the following is a soap (TV Programme)?","Power Rangers", "Eastenders", "BBC News", "MTV Base", 2],
["How many legs does a spider have?","4", "6", "8", "10", 3],
["What is the tallest animal in the world?","The giraffe", "Crocodile", "Bear", "Fox", 1]
                  ]
    #picks a random item from the list
    question = random.choice(questions)
    #removes last event
    pygame.event.clear()
    screen=pygame.display.set_mode((610, 650))
    questionmenu=pygame.image.load("images/menu/questionmenu.png")
    screen.blit(questionmenu,(0,0))
    #displays question + answers
    print(question[0],90,64)
    print(question[1],60,260)
    print(question[2],330,260)
    print(question[3],60,500)
    print(question[4],330,500)
    q=False
    choice = 0
    while not q:
        pygame.display.flip()
        ev = pygame.event.get()
        for event in ev:
            if event.type == pygame.MOUSEBUTTONUP:
                pos = pygame.mouse.get_pos()
                #detect which option is chosen
                if pos[0] in range(28,280) and pos[1] in range(238,328):
                    choice=1
                elif pos[0] in range(308,558) and pos[1] in range(238,328):
                    choice=2
                elif pos[0] in range(28,280) and pos[1] in range(475,562):
                    choice=3
                elif pos[0] in range(308,558) and pos[1] in range(475,562):
                    choice=4
                if choice:
                 if choice==question[5]:
                            #if correct option is chosen then Bool1 is returned
                     print("Correct! You go first!")
                     time.sleep(2)
                     return True
                 else:
                            #if correct option is not chosen then Bool2 is returned
                     print("That's not correct! You go second.")
                     time.sleep(2)
                     return False

if __name__=="__main__":
    askquestion()






###NEWFILE




"""
============================== TicTacPro ==============================
FILE: Logic.py
MODIFIED: 15/11/2015
STATUS: Complete
FILE DESCRIPTION:
The logic.py file has code used by multiple files of the program, a shared code file
such as the game logic, drawing the board, print functions etc
USAGE:
This is a client file as it is for the drawing and logic aspect of the game.
"""

from main import *
from settings import *

def print(text,x=40,y=615):
    """
    FUNCTION NAME: printcoord()
    PARAMETERS: 3
        text (string; mandatory):
        x (integer; optional"): The X co-ordinate of the top left corner of the text output
        y (integer; optional"): The Y co-ordinate of the top left corner of the text output

    FUNCTION DESCRIPTION:
    lets you draw text anywhere on the screen via the coordinates of the top left
    corner of the text displayed, if no coordinates are passed in it defaults to the bottom left of
    the screen which I have designated as the display bar section.
    """
    pygame.draw.rect(screen, (0,0,0), (10,610,600,40), 0)
    font = pygame.font.Font(None, 32)
    textg = font.render(str(text), 4, (255, 255, 255))
    screen.blit(textg, (x,y))
    pygame.display.flip()

def quitgame():
    """
    FUNCTION NAME: quitgame()
    PARAMETERS: 0
    FUNCTION DESCRIPTION:
        excecutes the correct procedure for closing the program.
    """
    pygame.quit()
    sys.exit()
```

```python
def gamewon(used):
    """
    FUNCTION NAME: gamewon()
    PARAMETERS: 1
        used (list; mandatory): The list representation of the board, either 1-9, "x" or "o".

    FUNCTION DESCRIPTION:
        Decides wether the game is won taking in the used table
        that indicates current game state and checking if there are 3 in a row
        """
    if used[0]==used[1] and used[1]==used[2]: return (True,1)
    elif used[3]==used[4] and used[4]==used[5]: return (True,2)
    elif used[6]==used[7] and used[7]==used[8]: return (True,3)
    elif used[0]==used[3] and used[3]==used[6]: return (True,4)
    elif used[1]==used[4] and used[4]==used[7]: return (True,5)
    elif used[2]==used[5] and used[5]==used[8]: return (True,6)
    elif used[0]==used[4] and used[4]==used[8]: return (True,7)
    elif used[2]==used[4] and used[4]==used[6]: return (True,8)
    else: return (False,999)

def validmove(pos,used):
    """
    FUNCTION NAME: validmove()
    PARAMETERS: 2
        pos (tuple; mandatory): A tuple containing the coordinate of a mouse click (0-620,0-690)
        used (list; mandatory): The list representation of the board, either 1-9, "x" or "o".

    FUNCTION DESCRIPTION:
        Checks the mouse position coordinates against the board "hitboxes" and returns a boolean
        defining wether the click was in a valid location of a box.
    """
    #rickageddon
    rickcheck(pos)

    if pos[0] in range(10,200) and pos[1] in range(10,200): sqr=0
    elif pos[0] in range(210,400) and pos[1] in range(10,200): sqr=1
    elif pos[0] in range(410,600) and pos[1] in range(10,200): sqr=2
    elif pos[0] in range(10,200) and pos[1] in range(210,400): sqr=3
    elif pos[0] in range(210,400) and pos[1] in range(210,400): sqr=4
    elif pos[0] in range(410,600) and pos[1] in range(210,400): sqr=5
    elif pos[0] in range(10,200) and pos[1] in range(410,600): sqr=6
    elif pos[0] in range(210,400) and pos[1] in range(410,600): sqr=7
    elif pos[0] in range(410,600) and pos[1] in range(410,600): sqr=8
    else:
        return False

    if used[sqr] not in ["o","x"]:
            return True
    else:
        return False

def rickcheck(pos):
    """super special move that rolls the screen"""
    if pos[0] in range(580,610) and pos[1] in range(610,640):
        cena=pygame.image.load("images/misc/cena.png")
        pygame.image.save(screen,"images/misc/temp.png")
        pygame.mixer.music.load("music/special.mp3")
        pygame.mixer.music.play(-1)
        posit=(-200)
        num=10
        while num<40:
            if posit>500:
                posit=(-200)
            screen.blit(rick,(posit,-50))
            screen.blit(rick,(posit+num,50))
            screen.blit(rick,(posit+num*2,150))
            screen.blit(rick,(posit+num*3,250))
            screen.blit(rick,(posit+num*4,350))
            if num>15:
                screen.blit(rick,(posit,-50))
                screen.blit(rick,(50,posit+num))
                screen.blit(rick,(150,posit+num*2))
                screen.blit(rick,(250,posit+num*3))
                screen.blit(rick,(350,posit+num*4))
            pygame.display.flip()
            num+=0.5
            posit+=20
            time.sleep(0.2)
            ev = pygame.event.get()
            for event in ev:
                if event.type == pygame.MOUSEBUTTONUP:
                    pos = pygame.mouse.get_pos()
                    screen.blit(rick,(pos[0]-50,pos[1]-50))
                    pygame.display.flip()
        pygame.mixer.music.pause()
        random.play()
        time.sleep(1.6)
        screen.blit(cena,(-350,-50))
        pygame.display.flip()
        time.sleep(4.4)
        pygame.mixer.music.unpause()

        temp=pygame.image.load("images/misc/temp.png")
        screen.blit(pygame.image.load("images/misc/temp.png"),(0,0))
        pygame.display.flip()

def drawbox(turn,pos,used,images):
    """
    FUNCTION NAME: drawbox()
    PARAMETERS: 4
        turn (boolean; mandatory): A boolean value that determines who's turn it is, True for x. False for o)
        pos (tuple; mandatory): A tuple containing the coordinate of a mouse click (0-620,0-690)
        used (list; mandatory): The list representation of the board, either 1-9, "x" or "o".
        images (list; mandatory): A list that stores the games current images for the board to use
         and it stored as objects in a list

    FUNCTION DESCRIPTION:
        Draws the game board, redraws it every time the board is generated. can handle different
        images sets thanks to the image list
    """
    #box7
    if pos[0] in range(10,200) and pos[1] in range(10,200) and used :
        if turn:
            screen.blit(images[1],(10,10))
            used[0]="x"
        else:
            screen.blit(images[2],(10,10))
            used[0]="o"
    #box8
    elif pos[0] in range(210,400) and pos[1] in range(10,200):
        if turn:
            screen.blit(images[1],(210,10))
            used[1]="x"
        else:
            screen.blit(images[2],(210,10))
            used[1]="o"
    #box9
    elif pos[0] in range(410,600) and pos[1] in range(10,200):
        if turn:
            screen.blit(images[1],(410,10))
            used[2]="x"
        else:
            screen.blit(images[2],(410,10))
            used[2]="o"
    #box4
    elif pos[0] in range(10,200) and pos[1] in range(210,400):
        if turn:
            screen.blit(images[1],(10,210))
            used[3]="x"
        else:
            screen.blit(images[2],(10,210))
            used[3]="o"
    #box5
    elif pos[0] in range(210,400) and pos[1] in range(210,400):
        if turn:
            screen.blit(images[1],(210,210))
            used[4]="x"
        else:
            screen.blit(images[2],(210,210))
            used[4]="o"
    #box6
    elif pos[0] in range(410,600) and pos[1] in range(210,400):
        if turn:
            screen.blit(images[1],(410,210))
            used[5]="x"
        else:
            screen.blit(images[2],(410,210))
            used[5]="o"
    #box1
    elif pos[0] in range(10,200) and pos[1] in range(410,600):
        if turn:
            screen.blit(images[1],(10,410))
            used[6]="x"
        else:
            screen.blit(images[2],(10,410))
            used[6]="o"
    #box2
    elif pos[0] in range(210,400) and pos[1] in range(410,600):
        if turn:
            screen.blit(images[1],(210,410))
            used[7]="x"
        else:
            screen.blit(images[2],(210,410))
            used[7]="o"
    #box3
    elif pos[0] in range(410,600) and pos[1] in range(410,600):
        if turn:
            screen.blit(images[1],(410,410))
            used[8]="x"
        else:
            screen.blit(images[2],(410,410))
            used[8]="o"

    turn=not turn
    pygame.display.flip()
    return used

def drawline(wincombo,turn,images):
    """
    FUNCTION NAME: drawbox()
    PARAMETERS: 3
        wincombo (integer; mandatory): An integer that represents what combination has been achieved
        turn (boolean; mandatory): A boolean value that determines who's turn it is, True for x. False for o)
        images (list; mandatory): A list that stores the games current images for the board to use,
        stored as objects in a list

    FUNCTION DESCRIPTION:
        When the game is won it creates the effect of the flashing winning boxes
    """
    xw=images[3]
    ow=images[4]
    x=images[1]
    o=images[2]
    #creates the flashing effect
    for count in range(0,6):
        #changes every other time
        if count/2==count//2:
            winsound.play()
            if wincombo==1:
                if turn:screen.blit(xw,(10,10)),screen.blit(xw,(210,10)),screen.blit(xw,(410,10))
                else:screen.blit(ow,(10,10)),screen.blit(ow,(210,10)),screen.blit(ow,(410,10))

            elif wincombo==2:
                if turn:screen.blit(xw,(10,210)),screen.blit(xw,(210,210)),screen.blit(xw,(410,210))
                else:screen.blit(ow,(10,210)),screen.blit(ow,(210,210)),screen.blit(ow,(410,210))

            elif wincombo==3:
                if turn:screen.blit(xw,(10,410)),screen.blit(xw,(210,410)),screen.blit(xw,(410,410))
                else:screen.blit(ow,(10,410)),screen.blit(ow,(210,410)),screen.blit(ow,(410,410))

            elif wincombo==4:
                if turn:screen.blit(xw,(10,10)),screen.blit(xw,(10,210)),screen.blit(xw,(10,410))
                else:screen.blit(ow,(10,10)),screen.blit(ow,(10,210)),screen.blit(ow,(10,410))

            elif wincombo==5:
                if turn:screen.blit(xw,(210,10)),screen.blit(xw,(210,210)),screen.blit(xw,(210,410))
                else:screen.blit(ow,(210,10)),screen.blit(ow,(210,210)),screen.blit(ow,(210,410))

            elif wincombo==6:
                if turn:screen.blit(xw,(410,10)),screen.blit(xw,(410,210)),screen.blit(xw,(410,410))
                else:screen.blit(ow,(410,10)),screen.blit(ow,(410,210)),screen.blit(ow,(410,410))

            elif wincombo==7:
                if turn:screen.blit(xw,(10,10)),screen.blit(xw,(210,210)),screen.blit(xw,(410,410))
                else:screen.blit(ow,(10,10)),screen.blit(ow,(210,210)),screen.blit(ow,(410,410))

            elif wincombo==8:
                if turn:screen.blit(xw,(410,10)),screen.blit(xw,(210,210)),screen.blit(xw,(10,410))
                else:screen.blit(ow,(410,10)),screen.blit(ow,(210,210)),screen.blit(ow,(10,410))
        else:
            if wincombo==1:
                if turn:screen.blit(x,(10,10)),screen.blit(x,(210,10)),screen.blit(x,(410,10))
                else:screen.blit(o,(10,10)),screen.blit(o,(210,10)),screen.blit(o,(410,10))

            elif wincombo==2:
                if turn:screen.blit(x,(10,210)),screen.blit(x,(210,210)),screen.blit(x,(410,210))
                else:screen.blit(o,(10,210)),screen.blit(o,(210,210)),screen.blit(o,(410,210))

            elif wincombo==3:
                if turn:screen.blit(x,(10,410)),screen.blit(x,(210,410)),screen.blit(x,(410,410))
                else:screen.blit(o,(10,410)),screen.blit(o,(210,410)),screen.blit(o,(410,410))

            elif wincombo==4:
                if turn:screen.blit(x,(10,10)),screen.blit(x,(10,210)),screen.blit(x,(10,410))
                else:screen.blit(o,(10,10)),screen.blit(o,(10,210)),screen.blit(o,(10,410))

            elif wincombo==5:
                if turn:screen.blit(x,(210,10)),screen.blit(x,(210,210)),screen.blit(x,(210,410))
                else:screen.blit(o,(210,10)),screen.blit(o,(210,210)),screen.blit(o,(210,410))

            elif wincombo==6:
                if turn:screen.blit(x,(410,10)),screen.blit(x,(410,210)),screen.blit(x,(410,410))
                else:screen.blit(o,(410,10)),screen.blit(o,(410,210)),screen.blit(o,(410,410))

            elif wincombo==7:
                if turn:screen.blit(x,(10,10)),screen.blit(x,(210,210)),screen.blit(x,(410,410))
                else:screen.blit(o,(10,10)),screen.blit(o,(210,210)),screen.blit(o,(410,410))

            elif wincombo==8:
                if turn:screen.blit(x,(410,10)),screen.blit(x,(210,210)),screen.blit(x,(10,410))
                else:screen.blit(o,(410,10)),screen.blit(o,(210,210)),screen.blit(o,(10,410))
        pygame.display.flip()
        time.sleep(0.5)

if __name__=="__main__":
    print("you need to run the main program")




###NEWFILE




"""
============================== TicTacPro ==============================
FILE: Main.py
MODIFIED: 15/11/2015
STATUS: Complete
FILE DESCRIPTION:
The Main.py file is the central state for the game, initilises the game and
all it's images and features, it also launches the splash screen when booted.
"""

#installing all the modules required for this game
try:
    module="Time"
    import time
    module="Sys"
    import sys
    module="Pygame"
    import pygame
    module="Math"
    import math
    module="os"
    import os
#if none of these work then it will throw an error and tell you what you need
except:
    print("Error - You don't have the required modules installed!")
    print("Please install Module '{0}'! ".format(module))
    for count in range(0,50000000):
        pass
    raise SystemExit

pygame.init()
#game wide initalistion, starts all the relevant aspects to the game and loads the standard images
pygame.display.set_icon(pygame.image.load("images/misc/icon.png"))
screen=pygame.display.set_mode((610, 650))
clock = pygame.time.Clock()
images=[
pygame.image.load("images/classic/board.png"),
pygame.image.load('images/classic/x.png'),
pygame.image.load('images/classic/o.png'),
pygame.image.load("images/classic/xwon.png"),
pygame.image.load("images/classic/owin.png")
]
rick=pygame.image.load("images/misc/rick.png")
pygame.display.set_caption("TicTacPro Game","TicTacPro")
difficulty="medium"
#all sound files from sounddogs.com royalty free and some editied by me
pygame.mixer.music.load("music/harder.mp3")
clicksound=pygame.mixer.Sound("music/fx/click.wav")
winsound=pygame.mixer.Sound("music/fx/win.wav")
losersound=pygame.mixer.Sound("music/fx/loser.wav")
nosound=pygame.mixer.Sound("music/fx/no.wav")
random=pygame.mixer.Sound("music/fx/random.wav")
mainmenuimg=pygame.image.load("images/menu/mainmenu.png")

#imports each file so the functions can be called and run in this program
try:
    from offline2p import *
    from offline1p import *
    from online import *
    from achievements import *
    from settings import *
    from firstgo import *
    from settings import *
    from logic import *
#if any files are missing will let you know what is missing
except ImportError:
    print("Error - You're missing game files!")
    print("Please download zip file again!")
    for count in range(0,50000000):
        pass
    raise SystemExit

def mainmenu(images, host="localhost", port=12341):
    """Runs the main menu, it opens the main menu, and allows you to access the rest of the game from here"""
    q = False
    while not q:
        pygame.event.clear
        screen=pygame.display.set_mode((610, 650))
        screen.blit(mainmenuimg,(0,0))
        pygame.draw.rect(screen, (0,0,0), [610, 0, 300, 700], 0)
        pygame.display.flip()
        ev = pygame.event.get()
        #if an event happens such as a keypress or mouse click it will run through this code
        for event in ev:
            if event.type == pygame.QUIT:
                quitgame()
            if event.type == pygame.MOUSEBUTTONUP:
                pos = pygame.mouse.get_pos()
                pygame.draw.rect(screen, (0,0,0), (10,610,600,40), 0)
                #checks whether the mouse click was on a button, which i have defined by coordinates
                if pos[0] in range(50,560) and pos[1] in range(90,180):
                    print("Under Construction!")
                    #onlineconnect() #enter host and port information to connect to; then call the whosturn function and th
                    #whosturn=chooseturn()
                    online(True, images, host, port)

                elif pos[0] in range(50,560) and pos[1] in range(190,285):
                    #runs the code that asks who goes first
                    whosturn=askquestion()
                    #launches the offline 1p state
                    offline1p(difficulty,whosturn,images)

                elif pos[0] in range(50,560) and pos[1] in range(290,385):
                    offline2p(True,images)

                elif pos[0] in range(50,560) and pos[1] in range(390,485):
                    possimages=settingmenu()
                    #if the stlye is changed then this code will run, changing what images the program uses
                    if possimages!=None:
                        images=possimages

                elif pos[0] in range(50,560) and pos[1] in range(490,585):
                    achievementsview()

                elif pos[0] in range(534,600) and pos[1] in range(15,73):
                    #quits the game and leaves the loop
                    quitgame()
                    q=True


if __name__=="__main__":
    #this generates the splash screen for the program, this is run after the loading code as splash screens are traditional
    pygame.display.set_icon(pygame.image.load("images/misc/icon.png"))
    os.environ['SDL_VIDEO_CENTERED'] = '1'
    sega=pygame.mixer.Sound("music/fx/sega.wav")
    sega.play()
    screen = pygame.display.set_mode((200,200),pygame.NOFRAME)
    pygame.Surface([640,480], pygame.SRCALPHA, 32)
    splash=pygame.image.load("images/misc/splash.png")
    screen.blit(splash,(-30,-200))
    pygame.display.flip()
    time.sleep(2)
    pygame.display.set_icon(pygame.image.load("images/misc/icon.png"))
    screen=pygame.display.set_mode((610, 650))
    if len(sys.argv) > 1: # If the game was started through the launcher, pass the host and port from the arguments variabl
        mainmenu(images, sys.argv[1], int(sys.argv[2]))
    else: # If the game was started by opening the main.py file, show the following message and exit the game.
        #print("Start the game using the TicTacProLauncher.py file.")
        #debug
        mainmenu(images, "localhost", 12341)
        time.sleep(5)
```

```
###NEWFILE




"""
============================== TicTacPro ==============================
FILE: Offline1p.py
MODIFIED: 15/11/2015
STATUS: Complete
FILE DESCRIPTION:
the Offline1p.py file is the one player state that has AI for deciding where
the computer should play and also how the computer should act depending on
the difficulty
"""

from main import *
from logic import *
from achievements import *
import random
import time
def AIwinNext(used): #This code checks if the computer can win the next move
    for i in range(len(used)):
        if used[i] != "x" or used[i] != "o":
            oldi = used[i]
            used[i] = "o"
            isgamewoncheck = gamewon(used)
            used[i] = oldi
            if isgamewoncheck:
                if i == 7:
                    return (10,10)
                elif i == 8:
                    return (210,400)
                elif i == 9:
                    return (410,600)
                elif i == 4:
                    return (10,200)
                elif i == 5:
                    return (210,400)
                elif i == 6:
                    return (410,600)
                elif i == 1:
                    return (10,200)
                elif i == 2:
                    return (210,400)
                elif i == 3:
                    return (410,600)
            else:
                return None

def AIblockNext(used): #This checks if the player could win the next move, the computer will block them
    for i in range(len(used)):
        if used[i] != "x" or used[i] != "o":
            oldi = used[i]
            used[i] = "x"
            isgamewoncheck = gamewon(used)
            used[i] = oldi
            if isgamewoncheck:
                if i == 7:
                    return (10,10)
                elif i == 8:
                    return (210,400)
                elif i == 9:
                    return (410,600)
                elif i == 4:
                    return (10,200)
                elif i == 5:
                    return (210,400)
                elif i == 6:
                    return (410,600)
                elif i == 1 :
                    return (10,200)
                elif i == 2:
                    return (210,400)
                elif i == 3:
                    return (410,600)
            else:
                return None

def offline1p(difficulty, turn, images):
    """runs the offline 1 player iteration of the game"""
    screen=pygame.display.set_mode((610, 650))
    pygame.mixer.music.play(-1)
    screen.blit(images[0],(0,0))
    used=[7,8,9,4,5,6,1,2,3]
    pygame.display.flip()
    count=0
    isgamewon=(False,9)
    while count<9 and not isgamewon[0]:
        ev = pygame.event.get()
        for event in ev:
            if turn:
                # user move
                if event.type == pygame.MOUSEBUTTONUP:
                    pos = pygame.mouse.get_pos()
                    clicksound.play()
                    valid=validmove(pos,used)
                    if valid:
                        drawbox(turn,pos,used,images)
                        isgamewon=gamewon(used)
                        count+=1
                        if isgamewon[0]:
                            drawline(isgamewon[1],turn,images)
                            break
                        turn=not turn
                    else:
                        print("That is not a valid move.")
                        nosound.play()
            elif not turn:
                # AI's move
                if difficulty == "easy":
                    pos = (random.randint(10,600), random.randint(10,600))
                elif difficulty == "medium":
                    pos = None
                    while pos is None:
                        pos = AIwinNext(used)
                        pos = AIblockNext(used)
                        pos = (random.randint(10,600), random.randint(10,600))
                elif difficulty == "hard":
                    #This code checks if the computer can win the next move
                    for i in range(len(used)):
                        if used[i] != "x" or used != "o":
                            oldi = used[i]
                            used[i] = "o"
                            isgamewon = gamewon(used)
                            if isgamewon:
                                if i == 7:
                                    pos = (10,10)
                                elif i == 8:
                                    pos = (210,400)
                                elif i == 9:
                                    pos = (410,600)
                                elif i == 4:
                                    pos = (10,200)
                                elif i == 5:
                                    pos = (210,400)
                                elif i == 6:
                                    pos = (410,600)
                                elif i == 1 :
                                    pos = (10,200)
                                elif i == 2:
                                    pos = (210,400)
                                elif i == 3:
                                    pos = (410,600)
                                drawbox(turn,pos,used,images)
                            else:
                                used[i] = oldi

                    #This checks if the player could win the next move, the computer will block them
                    for i in range(len(used)):
                        if used[i] != "x" or used != "o":
                            oldi = used[i]
                            used[i] = "x"
                            isgamewon = gamewon(used)
                            if isgamewon:
                                if i == 7:
                                    pos = (10,10)
                                elif i == 8:
                                    pos = (210,400)
                                elif i == 9:
                                    pos = (410,600)
                                elif i == 4:
                                    pos = (10,200)
                                elif i == 5:
                                    pos = (210,400)
                                elif i == 6:
                                    pos = (410,600)
                                elif i == 1 :
                                    pos = (10,200)
                                elif i == 2:
                                    pos = (210,400)
                                elif i == 3:
                                    pos = (410,600)
                                drawbox(turn,pos,used,images)
                            else:
                                used[i] = oldi

                    #The computer will check if the coners are free, it will take the corners
                    for i in [1,3,7,9]:
                        if used[i] != "x" or used != "o":
                            oldi = used[i]
                            used[i] = "o"
                            isgamewon = gamewon(used)
                            if isgamewon:
                                if i == 7:
                                    pos = (10,10)
                                elif i == 9:
                                    pos = (410,600)
                                elif i == 1 :
                                    pos = (10,200)
                                elif i == 3:
                                    pos = (410,600)
                                drawbox(turn,pos,used,images)
                            else:
                                used[i] = oldi

                    #This code allows the computer to check if the centre is free, it will take it
                    for i in [5]:
                        if used[i] != "x" or used != "o":
                            oldi = used[i]
                            used[i] = "o"
                            isgamewon = gamewon(used)
                            if isgamewon:
                                if i == 5:
                                    pos = (210,400)
                                drawbox(turn,pos,used,images)
                            else:
                                used[i] = oldi
                    print("hard")
                valid=validmove(pos,used)
                if valid:
                    clicksound.play()
                    drawbox(turn,pos,used,images)
                    isgamewon=gamewon(used)
                    count+=1
                    if isgamewon[0]:
                        drawline(isgamewon[1],turn,images)
                        break
                    turn=not turn
                else:
                    pass
    if isgamewon:
        if turn:
            winner="1"
            print("The Winner is Player {0}!".format(winner))
            time.sleep(2)
            achievements("won")
        else:
            winner="2"
            print("The Winner is Player {0}!".format(winner))
            time.sleep(2)
            achievements("lost")
    else:
        losersound.play()
        print("No One Won!")
        time.sleep(2)
        achievements("draw")
    pygame.mixer.music.fadeout(2000)
    time.sleep(2)


if __name__=="__main__":
    offline1p("medium",True,images)




###NEWFILE




from main import *
from logic import *
"""
============================== TicTacPro ==============================
FILE: Offline2p.py
MODIFIED: 15/11/2015
```

```
STATUS: Complete
FILE DESCRIPTION:
the Offline2p.py file is the 2 player offline state of the Tic Tac Toe game,
allowing 2 players to play against offline on a single system.
"""

def offline2p(turn,images):
    """runs the offline 2 player iteration of the game"""
    #sets up the screen for running the offline2p gamemode
    screen=pygame.display.set_mode((610, 650))
    pygame.mixer.music.play(-1)
    screen.blit(images[0],(0,0))
    used=[7,8,9,4,5,6,1,2,3]
    pygame.display.flip()
    count=0
    isgamewon=(False,9)
    while count<9 and not isgamewon[0]:
        #every other time this ramdomises
        ev = pygame.event.get()
        for event in ev:
            #checks where you click
            if event.type == pygame.MOUSEBUTTONUP:
                pos = pygame.mouse.get_pos()
                valid=validmove(pos,used)
                if valid:
                    #if the input is in a valid position then it will run this code
                    clicksound.play()
                    drawbox(turn,pos,used,images)
                    isgamewon=gamewon(used)
                    count+=1
                    #isgamewon is a tuple (bool,string of "player 1" or 2"
                    if isgamewon[0]:
                        drawline(isgamewon[1],turn,images)
                        break
                    turn=not turn
                else:
                    nosound.play()
                    print("NO")
            elif event.type == pygame.QUIT:
                quitgame()
    #this runs when the game is over, either the max number of moves have happened
    #or someone has won
    if isgamewon[0]:
        #works out who won, runs the corresponding code
        if turn:
            winner = "1"
            achievements("won")
        else:
            winner = "2"
            achievements("lost")
        print("The Winner is Player {0}!".format(winner))
    else:

        achievements("draw")
        losersound.play()
        print("No One Won!")
    pygame.mixer.music.fadeout(2000)
    time.sleep(2)

if __name__=="__main__":
    offline2p(True,images)




###NEWFILE



"""
============================== TicTacPro ==============================
FILE: server.py
MODIFIED: 23/11/2015
STATUS: debug
FILE DESCRIPTION:
The server.py file adds the functionality of playing multiplayer online games on a local network.
Events during the execution of the file will be logged to events.log.
USAGE:
The server.py file has to reside on the server computer to which clients will connect.
"""

import socket, select, logging

"""
CLASS NAME: servergo()
CLASS DESCRIPTION:
```

```
Initializes the server needed to host an online multiplayer game of TicTacPro. Makes use of a TCP/IP sockets for the client
This server acts as a bridge between the two clients, i.e. sends information from one to the other. No actions are performe
"""
class servergo():
    """
    FUNCTION NAME: __init__
    PARAMETERS: 2
                ipaddr (string; optional; defaults to "localhost"): the server IP address which will host client connection
                port (integer; optional; defaults to 12341): the socket port which clients will connect to
    FUNCTION DESCRIPTION:
    Initializes the server connection with the default parameters unless others are provided.
    """
    def __init__(self, ipaddr = "localhost", port = 12341):
        self.connections = []   # Declare a list that will hold all connections.
        self.server = socket.socket(socket.AF_INET, socket.SOCK_STREAM) # Create an IPv4 TCP socket.
        self.server.bind((ipaddr, port))    # Bind the socket to the provided host and port.
        self.server.listen(2)   # Accept a maximum of 2 client connections.
        print("TicTacPro server started on host " + str(socket.gethostbyname(socket.gethostname())) + " and port " + str(po
        print("")

    """
    FUNCTION NAME: shutdown()
    PARAMETERS: 0
    FUNCTION DESCRIPTION:
    Closes the connection to each client and the completely shuts down the server. No connection to the server can be estab
    """
    def shutdown(self):
        for conn in self.connections:
            conn.close()
        self.server.shutdown(1)
        self.server.close()

    """
    FUNCTION NAME: play()
    PARAMETERS: 0
    FUNCTION DESCRIPTION:
    Accepts new connections and appends them to the connection list (self.connections).
    Transfers information from one client to the other.
    """
    def play(self):
        read, write, error = select.select(self.connections+[self.server], self.connections, self.connections, 0) # Get the

        for conn in read:
            if conn is self.server: # Handle new clients connecting to the server.
                c, addr = conn.accept()
                self.connections.append(c)
                print("New client connected: " + str(addr))
            else:
                data = conn.recv(1024)
                if not data: # If no data is received, the client has disconnected. Close the connection and end the online
                    conn.close()
                    self.shutdown()
                elif data: # If data is received, send it to the other client.
                    for client in self.connections:
                        if client != conn:
                            try:
                                client.send(data)
                            except Exception as e:
                                print("Could not send to " + str(conn))
                                logging.error("Could not send to " + str(conn) + " Exception details: " + str(e))

if __name__ == "__main__":
    logging.basicConfig(filename="events.log", format="%(asctime)s [server.py] %(message)s", datefmt="%Y/%m/%d %I:%M:%S %p"
    try:
        server = servergo()
        while True: # Keep server alive.
            server.play()
    except Exception as e:
        logging.exception(str(e))
    finally:
        server.shutdown()



###NEWFILE



"""
============================== TicTacPro ==============================
FILE: Online.py
MODIFIED: 15/11/2015
STATUS: Complete
FILE DESCRIPTION:
```

```python
the Online.py file is the online version of the tictactoe game, it allows
communication between 2 machines through a server, which sends commands for
not only board positions but also the chat, which allows text communication
between the 2 clients
"""
from main import *
from logic import *
from chat import *
from firstgo import *
import socket, select, pickle, logging

class GameClient():
    def __init__(self, host="localhost", port=12341):
        self.client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.client.connect((host, port))
        self.messages = []
        self.recvmessages = []

    def shutdown(self):
        self.client.shutdown(1)
        self.client.close()

    def send_message(self, msg):
        msg = pickle.dumps(msg)
        self.messages.append(msg)

    def recv_message(self):
        if self.recvmessages:
            return pickle.loads(self.recvmessages.pop(0))

    def poll(self):
        read, write, error = select.select ( [self.client], [self.client], [self.client], 0 )

        for conn in read:
            newmsg = self.client.recv(1024)
            if newmsg:
                self.recvmessages.append(newmsg)

        for conn in write:
            for i in self.messages:
                conn.send(i)
                self.messages.remove(i)

def online(turn, images, host, port):
    used = [7, 8, 9, 4, 5, 6, 1, 2, 3]
    count = 0
    isgamewon = (False, 9)
    chat=[]
    displaystring=""
    try:
        gamecli = GameClient(host, port)

        myturn = None
        while True:
            if myturn is None:
                myturn = askquestion()
                gamecli.send_message(myturn)
            gamecli.poll() # Send and receive messages between opponents.
            newmsg = gamecli.recv_message()
            if myturn is not None and newmsg is not None:
                if myturn == newmsg:
                    print("Both players answered correctly. Try again.")
                    myturn = None
                else:
                    break
        screen = pygame.display.set_mode((900, 650))
        pygame.mixer.music.play(-1)
        screen.blit(images[0], (0, 0))
        pygame.display.flip()

        while True:
            ev = pygame.event.get()
            for event in ev:
                if turn and event.type == pygame.MOUSEBUTTONUP:
                    pos = pygame.mouse.get_pos()
                    clicksound.play()
                    valid = validmove(pos, used)
                    if valid:
                        gamecli.send_message(pos)
                        drawbox(turn, pos, used, images)
                        isgamewon = gamewon(used)
                        count += 1
                        if isgamewon[0]:
                            drawline(isgamewon[1], turn, images)
                            break
                        turn = not turn
                    else:
                        print("That is not a valid move.")
                        nosound.play()

                elif event.type == pygame.KEYDOWN:
                    if event.key == 13:
                        if displaystring!="":
                            chat.insert(0,("You",displaystring))
                            gamecli.send_message(displaystring)
                        displaystring=""
                        drawlist(chat)
                    else:
                        displaystring,chat=chatinput(event.key,displaystring,chat)

                elif event.type == pygame.QUIT:
                    quitgame()

            gamecli.poll() # Send and receive messages between opponents.
            newmsg = gamecli.recv_message()

            if not turn and isinstance(newmsg, tuple):
                drawbox(turn, newmsg, used, images)
                clicksound.play()
                isgamewon = gamewon(used)
                count += 1
                if isgamewon[0]:
                    drawline(isgamewon[1], turn, images)
                    break
                turn = not turn

            if isinstance(newmsg, str):
                chat.insert(0,("Opponent",newmsg))
                drawlist(chat)
            if count == 9:
                break

        if isgamewon:
            if turn:
                winner="1"
                achievements("won")
            else:
                winner="2"
                achievements("lost")
            print("The Winner is Player {0}!".format(winner))
        else:
            losersound.play()
            achievements("draw")
            print("No One Won!")
        pygame.mixer.music.fadeout(2000)
        time.sleep(2)
    finally:
        gamecli.shutdown()

if __name__=="__main__":
    online(True, images, "localhost", 12341)



###NEWFILE



"""
============================== TicTacPro ==============================
FILE: Settings.py
MODIFIED: 15/11/2015
STATUS: Complete
FILE DESCRIPTION:
the Settings.py file is the state in which the program is changing visual
and audio settings about the game, such as the style or the music or the
difficulty.
"""
from main import *

#0-board
#1-x
#2-o
#3-xwin
#4-owin

def print(text):
    """changes the print function to now print to the gui 'status bar' """
    pygame.draw.rect(screen, (0,0,0), (10,610,600,40), 0)
    font = pygame.font.Font(None, 32)
    textg = font.render(str(text), 4, (255, 255, 255))
    screen.blit(textg, (40,610))
    pygame.display.flip()
#normal images

def settingmenu():
    settingsmenu=pygame.image.load("images/menu/settings/settingsmenu.png")
    q=False
    while not q:
        #display menu screen
        screen.blit(settingsmenu,(0,0))
        pygame.display.flip()
        ev = pygame.event.get()
        #EDP
        for event in ev:
            if event.type == pygame.MOUSEBUTTONUP:
                #pos is the coords for the mouse click
                pos = pygame.mouse.get_pos()
                #chooses which menu opens
                if pos[0] in range(30,200) and pos[1] in range(160,300):
                    images=stylemenu()
                elif pos[0] in range(400,600) and pos[1] in range(160,300):
                    songmenu()
                elif pos[0] in range(23,204) and pos[1] in range(540,600):
                    #if images has been defined it will return it if not it will
                    #return nothing and the game will continue to use whatever
                    #images are already loaded
                    try:
                        return images
                    except:
                        return None


def songmenu():
    songmenu=pygame.image.load("images/menu/settings/songmenu.png")
    screen.blit(songmenu,(0,0))
    pygame.display.flip()
    q=False
    while not q:
        ev = pygame.event.get()
        for event in ev:
            if event.type == pygame.MOUSEBUTTONUP:
                pos = pygame.mouse.get_pos()

                if pos[0] in range(30,200) and pos[1] in range(160,300):
                    print("Music Changed - Harder!")
                    pygame.mixer.music.load("music/harder.mp3")
                elif pos[0] in range(400,600) and pos[1] in range(160,300):
                    print("Music Changed - Lucky!")
                    pygame.mixer.music.load("music/lucky.mp3")
                elif pos[0] in range(30,200) and pos[1] in range(370,510):
                    print("Music Changed - Nostalgia!")
                    pygame.mixer.music.load("music/oldschool.mp3")
                elif pos[0] in range(400,600) and pos[1] in range(370,510):
                    print("Music Changed - Oh, Christmas Tree!")
                    pygame.mixer.music.load("music/christmas.mp3")
                elif pos[0] in range(23,204) and pos[1] in range(540,600): # Back button.
                    q=True
def stylemenu():
    stylesmenu=pygame.image.load("images/menu/settings/stylemenu.png")
    screen.blit(stylesmenu,(0,0))
    pygame.display.flip()
    q=False
    while not q:
        ev = pygame.event.get()
        for event in ev:
            if event.type == pygame.MOUSEBUTTONUP:
                pos = pygame.mouse.get_pos()
            #top left button

                if pos[0] in range(30,200) and pos[1] in range(160,300):
                    print("Board Loaded - Classic Board")
                    #the game pulls images from the list, change the content
                    #of the list, the images used changes, for ease of use
                    images=[
                    pygame.image.load("images/classic/board.png"),
                    pygame.image.load('images/classic/x.png'),
                    pygame.image.load('images/classic/o.png'),
                    pygame.image.load("images/classic/xwon.png"),
                    pygame.image.load("images/classic/owin.png")
                    ]

                elif pos[0] in range(400,600) and pos[1] in range(160,300):
                    print("Board Loaded - Test Board")
                    images=[
                    pygame.image.load("images/test/board.png"),
                    pygame.image.load('images/test/x.png'),
                    pygame.image.load('images/test/o.png'),
                    pygame.image.load("images/test/xwon.png"),
                    pygame.image.load("images/test/owin.png")
                    ]

                elif pos[0] in range(30,200) and pos[1] in range(370,510):
                    print("Board Loaded - Christmas Board")
                    images=[
                    pygame.image.load("images/christmas/board.png"),
                    pygame.image.load('images/christmas/x.png'),
                    pygame.image.load('images/christmas/o.png'),
                    pygame.image.load("images/christmas/xwon.png"),
                    pygame.image.load("images/christmas/owin.png")
                    ]

                elif pos[0] in range(400,600) and pos[1] in range(370,510):
                    print("Board Loaded - Mushroom Board")
                    images=[
                    pygame.image.load("images/mushroom/board.png"),
                    pygame.image.load('images/mushroom/x.png'),
                    pygame.image.load('images/mushroom/o.png'),
                    pygame.image.load("images/mushroom/xwon.png"),
                    pygame.image.load("images/mushroom/owin.png")
                    ]

                elif pos[0] in range(23,204) and pos[1] in range(540,600):
                    try:
                        return images
                    except:
                        return None

if __name__=="__main__":
    settingmenu()




###NEWFILE


"""
============================== TicTacPro ==============================
FILE: TicTacProLauncher.py
MODIFIED: 15/11/2015
STATUS: Complete
FILE DESCRIPTION:
the TicTacToeLauncher.py file is the bootstrap for the game, it is used to get
the server and port if you wish to launch the online version of the game.
"""

import os, time, sys

def valYN(prompt):
    while True:
        value = input(prompt).upper()
        if value not in ["Y", "N"]:
            print("Please enter Y or N according to your answer.")
            continue
        else:
            return value

def valhost(prompt):
    while True:
        value = input(prompt)
        if " " in value:
            print("Host cannot contain spaces.")
            continue
        else:
            return value

def valport(prompt):
    while True:
        try:
            value = int(input(prompt))
            break
        except ValueError:
            print("Insert numbers only.")
            continue
    return str(value) # Convert to string in order to pass as sys argument.

def startGame(host, port):
        if sys.version_info[0] < 3:
            try:
                os.system("python3 main.py " + host + " " + port) # If the computer has both Python 2 and Python 3 installe
            except:
                raise OSError("Please install Python 3 to play the game.")
        elif sys.version_info[0] == 3:
            os.system("python3 main.py " + host + " " + port)

if __name__ == "__main__":
    print("=========================== Welcome to TicTacPro! ===========================")
    print("We are preparing the TicTacPro GUI for you.")
    ans1 = valYN("Before we start, we need to know whether you will want to play online. (Y/N)")
    if ans1 == "Y":
        print()
        print("In order to play in online mode please provide the host name and the port number you will be connecting to."
        host = valhost("Host: ")
        port = valport("Port: ")

        print()
        print("TicTacPro will now initiate with enabled online multiplayer on host " + host + " and port " + port)
        time.sleep(3)
        startGame(host, port)
    elif ans1 == "N":
        print("You have chosen to not play online during this session.")
        print("If you choose the online mode from the main menu, an exception will be raised.")
        startGame("localhost", "12341")




###NEWFILE




"""
-------------Intial File--------------

This is a file built by Jiminy Haynes before finalising the plan for the
TicTacToe game, it is a line based GUI and was made to test concepts
and understand the game mechanics better
"""

import time
import msvcrt
import os
os.system("mode con cols=50 lines=28")
def draw_board(box):
    print("-"*49)
    for count2 in range(0,3):
        for count in range(0,7):
            print("|{0}|{1}|{2}|".format(box[count2][0][count],box[count2][1][count],box[count2][2][count]))
        print("-"*49)

def generate_X():
    x0=(" x           x ")
    x1=("   x       x   ")
    x2=("     x   x     ")
    x3=("       x       ")
    x4=("     x   x     ")
    x5=("   x       x   ")
    x6=(" x           x ")
    x=[x0,x1,x2,x3,x4,x5,x6]
    return x

def generate_O():
    o0=("       x       ")
    o1=("     x   x     ")
    o2=("   x       x   ")
    o3=(" x           x ")
    o4=("   x       x   ")
    o5=("     x   x     ")
    o6=("       x       ")
    o=[o0,o1,o2,o3,o4,o5,o6]
    return o

def generate_B():
    b0=("               ")
    b1=("               ")
    b2=("               ")
    b3=("               ")
    b4=("               ")
    b5=("               ")
    b6=("               ")
    b=[b0,b1,b2,b3,b4,b5,b6]
    return b
box_x=generate_X()
box_o=generate_O()
box_b=generate_B()

def generate_board(board,command):
```

```python
    #command 0 = player
    #command 1 = y axis
    #command 2 = x axis

    if command[0]=="x":
        board[command[1]][command[2]]=box_x
    elif command[0]=="o":
        board[command[1]][command[2]]=box_o
    draw_board(board)

def intial_generate_board():
    line1=[box_b,box_b,box_b]
    line2=[box_b,box_b,box_b]
    line3=[box_b,box_b,box_b]

    board=[line1,line2,line3]
    draw_board(board)
    return board


def get_user_move():
    valid=False
    while not valid:
        try:
            #user_move=int(input())
            user_move=int(msvcrt.getch())
            if user_move==7:
                move="a1"
            elif user_move==8:
                move="a2"
            elif user_move==9:
                move="a3"
            elif user_move==4:
                move="b1"
            elif user_move==5:
                move="b2"
            elif user_move==6:
                move="b3"
            elif user_move==1:
                move="c1"
            elif user_move==2:
                move="c2"
            elif user_move==3:
                move="c3"
            valid=True
        except ValueError:
            print("That's not a valid input!")
    return move

def generate_command(board,countplayer):
    while countplayer!=0:
        if countplayer//2==countplayer/2:
            print("Player Two (X)!")
            player="x"
        else:
            print("Player One (O)!")
            player="o"
        countplayer-=1
        valid=False
        while not valid:
            coord=get_user_move()

            if coord[0].lower()=="a":
                yaxis=0
            elif coord[0].lower()=="b":
                yaxis=1
            elif coord[0]=="c":
                yaxis=2
            else:
                print("FAIL")
            command=(player,yaxis,(int(coord[1])-1))
            valid=True
            if board[command[1]][command[2]][0]!=("               "):
                print("That is not a valid move!")
                valid=False
        return command,countplayer

def generate_combos():
    combo1=["00","01","02"]
    combo2=["00","10","20"]
    combo3=["02","12","22"]
    combo4=["20","21","22"]
    combo5=["01","11","21"]
    combo6=["10","11","12"]
    combo7=["00","11","22"]
    combo8=["02","11","20"]
    combos=[combo1,combo2,combo3,combo4,combo5,combo6,combo7,combo8]
    return combos

def game_won(gamewon,combos,board):
    for count in range(0,8):

        if (
            board [int(combos[count][0][0])] [int(combos[count][0][1])] [0] == board [int(combos[count][1][0])] [int(combos
            board [int(combos[count][1][0])] [int(combos[count][1][1])] [0] == board [int(combos[count][2][0])] [int(combos
            board [int(combos[count][2][0])] [int(combos[count][2][1])] [0] == (" x           x ")):

                gamewon=("Player 2",True)

        elif(
            board [int(combos[count][0][0])] [int(combos[count][0][1])] [0] == board [int(combos[count][1][0])] [int(combos
            board [int(combos[count][1][0])] [int(combos[count][1][1])] [0] == board [int(combos[count][2][0])] [int(combos
            board [int(combos[count][2][0])] [int(combos[count][2][1])] [0] == ("       x       ")):

                gamewon=("Player 1",True)

    return gamewon
def winstate(gamewon,board):
    draw_board(board)
    q=True
    print()
    print("{0} Has won!".format(gamewon[0]))
    time.sleep(3)
    return q

def losestate(board):
    draw_board(board)
    print()
    print("No one won!")
    time.sleep(3)
    q=True
    return q

def main():
    q=False
    countplayer=9
    combos=generate_combos()
    gamewon=("No One",False)
    board=intial_generate_board()
    print("Welcome to Tic Tac Pro!")
    while not q:
        if countplayer!=0:
            command,countplayer=generate_command(board,countplayer)
            generate_board(board,command)
            gamewon=game_won(gamewon,combos,board)
            print()
            if gamewon[1]==True:
                q=winstate(gamewon,board)

        else:
            gamewon=game_won(gamewon,combos,board)
            if gamewon[1]==True:
                q=winstate(gamewon,board)
            else:
                q=losestate(board)
    print("Thanks for playing!")
    time.sleep(3)
main()
```