University of Jeddah
College of Computer Science and Engineering
Department of Computer Science and Artificial Intelligence
Fundamentals of AI| CCAI-221

2 2 MATCHING NUMBERS GAME

prepared for: I.Amatulrahman Al-Subhi

prepared by:

Fai Al-Meganni - 2110006

Layan Al-Saud - 2110946

Joud AL-Baiti - 2111489

Shahad Al-Zahrani - 2112039

CCAI211: Fundamental of AI

College of Computer Science and Engineering, University of Jeddah

Idea description:

We have 10 cards flipped-down each 2 of them holds the same number, the game idea is to choose 2 cards as pairs and they must be identical to each other to score points. The game will be played by 2 competitors, the computer bot against the user, and whoever gets the highest score will win the game.

The problem:

Playing this kind of games virtually sometimes is boring and none competitive, but what if we played the game against smart and hard to beat competitor?

The solution:

We developed a nonboring game, the user against the computer bot. we have a board filled with 10 flipped-down cards contain numbers, in each round the player will choose 2 random positions, if they are identical the player will get the point of the round and the turn will be switched to the bot, if it was not identical, the player will not get a point and the turn will go to the bot, and so on. The game will resume until it's all matching cards are found and flipped up and got a winner. The player will try to win the game using their concentration but the bot will try to beat the other player using minmax algorithm to decide which card to flip.

Ai approach used to solve the problem:

For this report, we will concentrate on a technique termed the min-max algorithm to calculate the best achievable payoff against the best play. This approach starts by generating a game tree completely and then determining the utility of each terminal state and propagating the utility values upward in the tree by applying MIN and MAX operators on the nodes in the current level then at the root node using the minimax decision to select the move with the max (of the min) utility value. We used this approach for this game because the winner is determined by the number of points. With each movement that matches the cards correctly, the points will increase for the player.

Theoretical background:

To play this game with appropriate quality, we consulted sources for the rules of the game and how to write it as a programming code. So, we did write a programming code using python language.

How to Play the matching numbers game?

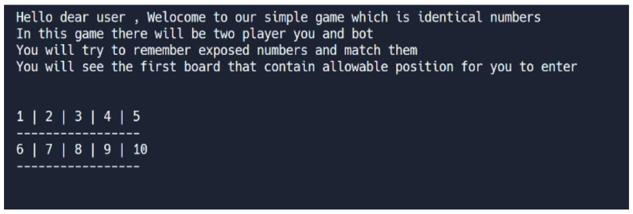
- The game starts with all the cards being shuffled well and flipped down across the table/board.
- Players take turns to turn over two cards, searching for the matching pair of numeral cards.
- If the cards selected are a pair, the player gets a point. If they are not the same, the cards are turned back over to rest in the same location on the board, and the play moves to the next player.
- Once all cards have been matched, the player with the most points wins.

The resources that helped us finding the game rules:

- https://en.wikipedia.org/wiki/Concentration_(card_game)
- https://www.ducksters.com/games/concentration_rules.php
- https://bicyclecards.com/how-to-play/concentration/

How to play the game:

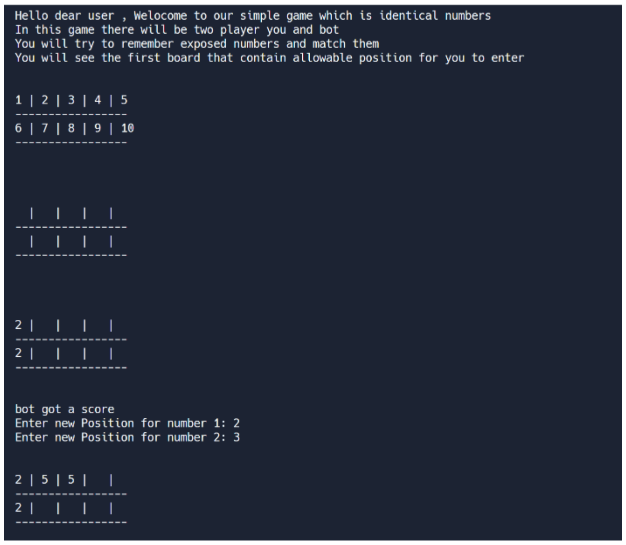
once you start the game…
a welcoming massage and the instruction
of the game will appear.



```
Hello dear user , Welocome to our simple game which is identical numbers
In this game there will be two player you and bot
You will try to remember exposed numbers and match them
You will see the first board that contain allowable position for you to enter


1 | 2 | 3 | 4 | 5
-----------------
6 | 7 | 8 | 9 | 10
-----------------
```

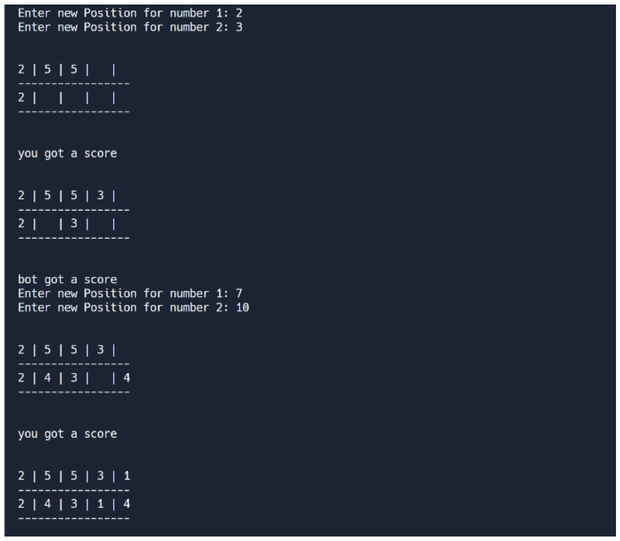
At the beginning…

an empty bored will appear, and the computer bot will
start playing the first round then the player.



```
Hello dear user , Welocome to our simple game which is identical numbers
In this game there will be two player you and bot
You will try to remember exposed numbers and match them
You will see the first board that contain allowable position for you to enter


1 | 2 | 3 | 4 | 5
-----------------
6 | 7 | 8 | 9 | 10
-----------------



  |   |   |   |
-----------------
  |   |   |   |
-----------------



2 |   |   |   |
-----------------
2 |   |   |   |
-----------------


bot got a score
Enter new Position for number 1: 2
Enter new Position for number 2: 3


2 | 5 | 5 |   |
-----------------
2 |   |   |   |
-----------------
```

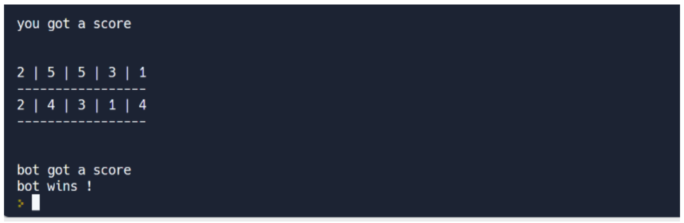
The game will resume round by round, until
all cards are flipped up



```
Enter new Position for number 1: 2
Enter new Position for number 2: 3


2 | 5 | 5 |   |
-----------------
2 |   |   |   |
-----------------


you got a score


2 | 5 | 5 | 3 |
-----------------
2 |   | 3 |   |
-----------------


bot got a score
Enter new Position for number 1: 7
Enter new Position for number 2: 10


2 | 5 | 5 | 3 |
-----------------
2 | 4 | 3 |   | 4
-----------------


you got a score


2 | 5 | 5 | 3 | 1
-----------------
2 | 4 | 3 | 1 | 4
-----------------
```

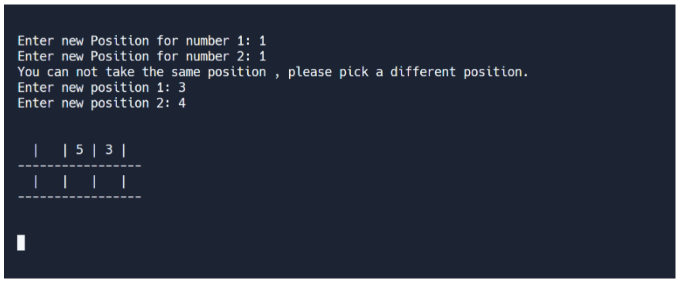
At the end of the game…

Whoever gets the highest number of points will win.
In this game the bot won the game.



```
you got a score


2 | 5 | 5 | 3 | 1
-----------------
2 | 4 | 3 | 1 | 4
-----------------



bot got a score
bot wins !
> 
```

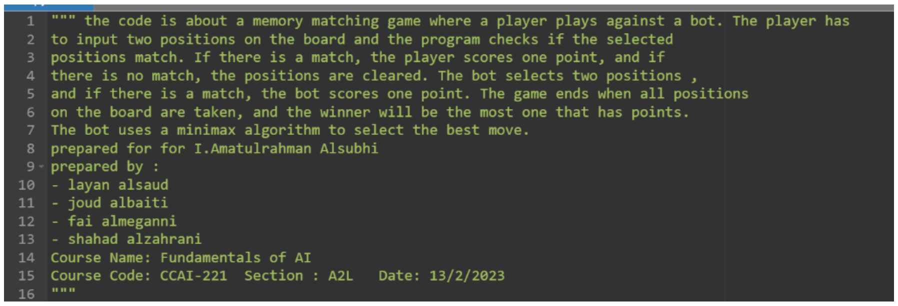
If the player picked a wrong position whether a wrong value or an already taken position, an error massage will appear to alert him and giving him another chance.



```
Enter new Position for number 1: 1
Enter new Position for number 2: 1
You can not take the same position , please pick a different position.
Enter new position 1: 3
Enter new position 2: 4


  |   | 5 | 3 |
----------------
  |   |   |   |
----------------

```

The code:

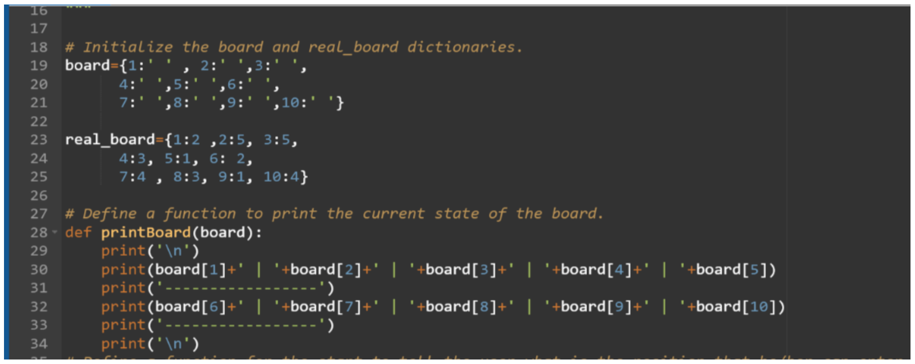
Code overview



```
""" the code is about a memory matching game where a player plays against a bot. The player has
to input two positions on the board and the program checks if the selected
positions match. If there is a match, the player scores one point, and if
there is no match, the positions are cleared. The bot selects two positions ,
and if there is a match, the bot scores one point. The game ends when all positions
on the board are taken, and the winner will be the most one that has points.
The bot uses a minimax algorithm to select the best move.
prepared for for I.Amatulrahman Alsubhi
prepared by :
- layan alsaud
- joud albaiti
- fai almeganni
- shahad alzahrani
Course Name: Fundamentals of AI
Course Code: CCAI-221  Section : A2L   Date: 13/2/2023
"""
```

We initialized an empty board and another board filled with numbers.

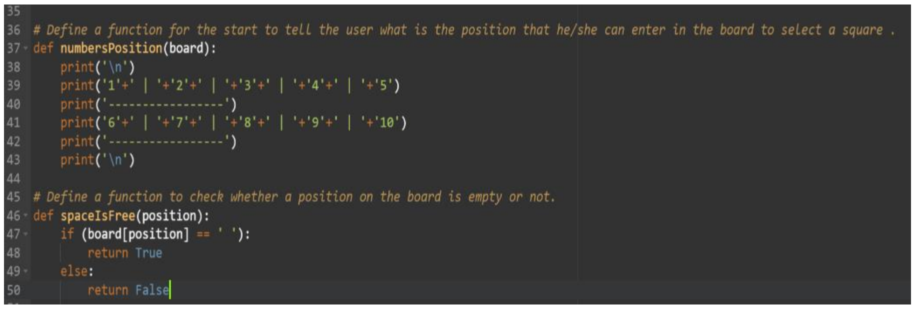
Then we defined a function to print the last updated board.



```

# Initialize the board and real_board dictionaries.
board={1:' ' , 2:' ',3:' ',
       4:' ',5:' ',6:' ',
       7:' ',8:' ',9:' ',10:' '}

real_board={1:2 ,2:5, 3:5,
       4:3, 5:1, 6: 2,
       7:4 , 8:3, 9:1, 10:4}

# Define a function to print the current state of the board.
def printBoard(board):
    print('\n')
    print(board[1]+' | '+board[2]+' | '+board[3]+' | '+board[4]+' | '+board[5])
    print('-----------------')
    print(board[6]+' | '+board[7]+' | '+board[8]+' | '+board[9]+' | '+board[10])
    print('-----------------')
    print('\n')
```

The numbersPosition(board) function is to inform the user about the allowable positions.

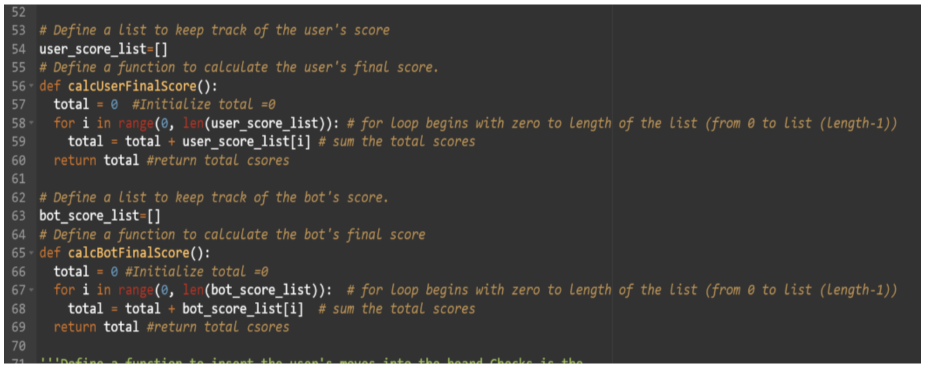
The spaceIsFree(position) function is to check whether a position on the board is empty or not.



```

# Define a function for the start to tell the user what is the position that he/she can enter in the board to select a square .
def numbersPosition(board):
    print('\n')
    print('1'+' | '+'2'+' | '+'3'+' | '+'4'+' | '+'5')
    print('----------------')
    print('6'+' | '+'7'+' | '+'8'+' | '+'9'+' | '+'10')
    print('----------------')
    print('\n')

# Define a function to check whether a position on the board is empty or not.
def spaceIsFree(position):
    if (board[position] == ' '):
        return True
    else:
        return False
```

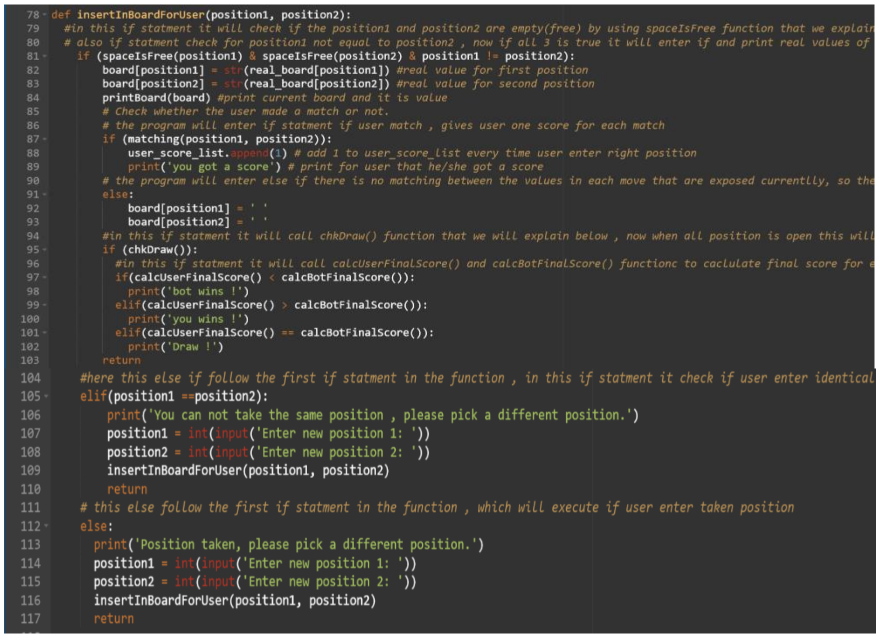
We defined a list for the computer bot and the player to store the number of correct guesses.

Then we created calcUserFinalScore() and calcBotFinalScore() functions to calculates the earned points in every round.



```python

# Define a list to keep track of the user's score
user_score_list=[]
# Define a function to calculate the user's final score.
def calcUserFinalScore():
  total = 0  #Initialize total =0
  for i in range(0, len(user_score_list)): # for loop begins with zero to length of the list (from 0 to list (length-1))
    total = total + user_score_list[i] # sum the total scores
  return total #return total csores

# Define a list to keep track of the bot's score.
bot_score_list=[]
# Define a function to calculate the bot's final score
def calcBotFinalScore():
  total = 0 #Initialize total =0
  for i in range(0, len(bot_score_list)):  # for loop begins with zero to length of the list (from 0 to list (length-1))
    total = total + bot_score_list[i]  # sum the total scores
  return total #return total csores

'''Define a function to insert the user's moves into the board Checks is the
```

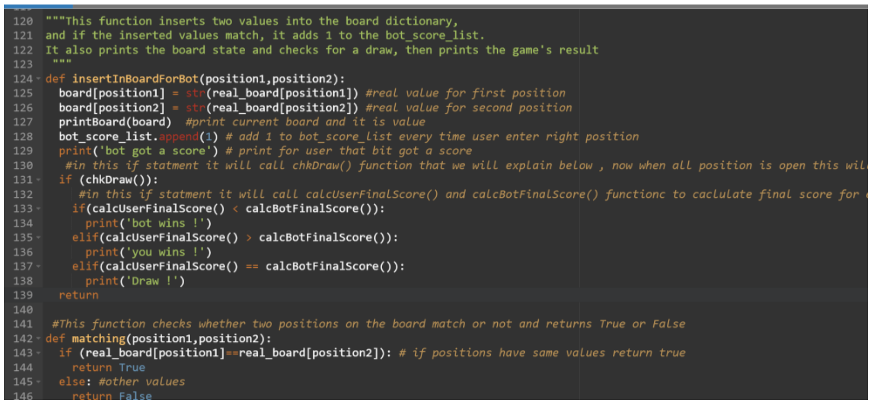
Defining a function to insert the user's moves into the board and checks if the position is free or not also gives a score and check who wins.



```python
def insertInBoardForUser(position1, position2):
  #in this if statment it will check if the position1 and position2 are empty(free) by using spaceIsFree function that we explain
  # also if statment check for position1 not equal to position2 , now if all 3 is true it will enter if and print real values of
    if (spaceIsFree(position1) & spaceIsFree(position2) & position1 != position2):
        board[position1] = str(real_board[position1]) #real value for first position
        board[position2] = str(real_board[position2]) #real value for second position
        printBoard(board) #print current board and it is value
        # Check whether the user made a match or not.
        # the program will enter if statment if user match , gives user one score for each match
        if (matching(position1, position2)):
            user_score_list.append(1) # add 1 to user_score_list every time user enter right position
            print('you got a score') # print for user that he/she got a score
        # the program will enter else if there is no matching between the values in each move that are exposed currentlly, so the
        else:
            board[position1] = ' '
            board[position2] = ' '
        #in this if statment it will call chkDraw() function that we will explain below , now when all position is open this will
        if (chkDraw()):
          #in this if statment it will call calcUserFinalScore() and calcBotFinalScore() functionc to caclulate final score for e
          if(calcUserFinalScore() < calcBotFinalScore()):
            print('bot wins !')
          elif(calcUserFinalScore() > calcBotFinalScore()):
            print('you wins !')
          elif(calcUserFinalScore() == calcBotFinalScore()):
            print('Draw !')
        return
    #here this else if follow the first if statment in the function , in this if statment it check if user enter identical
    elif(position1 ==position2):
        print('You can not take the same position , please pick a different position.')
        position1 = int(input('Enter new position 1: '))
        position2 = int(input('Enter new position 2: '))
        insertInBoardForUser(position1, position2)
        return
    # this else follow the first if statment in the function , which will execute if user enter taken position
    else:
      print('Position taken, please pick a different position.')
      position1 = int(input('Enter new position 1: '))
      position2 = int(input('Enter new position 2: '))
      insertInBoardForUser(position1, position2)
      return
```

The function insertInBoardForBot(position1,position2) is to inserts two values into the board dictionary, and if the inserted values match, it adds 1 to the bot_score_list. It also prints the board state and checks for a draw, then prints the game's result.

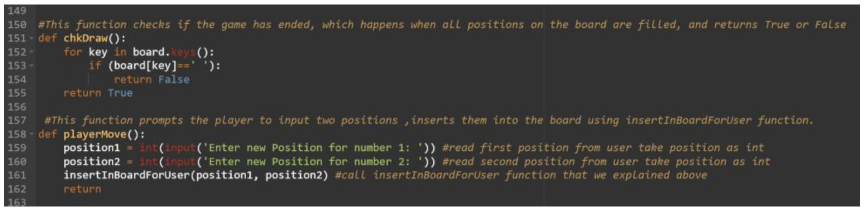
The function matching(position1,position2) checks whether two positions on the board match or not and returns True or False.



```python
"""This function inserts two values into the board dictionary,
and if the inserted values match, it adds 1 to the bot_score_list.
It also prints the board state and checks for a draw, then prints the game's result
 """
def insertInBoardForBot(position1,position2):
  board[position1] = str(real_board[position1]) #real value for first position
  board[position2] = str(real_board[position2]) #real value for second position
  printBoard(board)  #print current board and it is value
  bot_score_list.append(1) # add 1 to bot_score_list every time user enter right position
  print('bot got a score') # print for user that bit got a score
   #in this if statment it will call chkDraw() function that we will explain below , now when all position is open this wil
  if (chkDraw()):
     #in this if statment it will call calcUserFinalScore() and calcBotFinalScore() functionc to caclulate final score for 
    if(calcUserFinalScore() < calcBotFinalScore()):
      print('bot wins !')
    elif(calcUserFinalScore() > calcBotFinalScore()):
      print('you wins !')
    elif(calcUserFinalScore() == calcBotFinalScore()):
      print('Draw !')
  return

 #This function checks whether two positions on the board match or not and returns True or False
def matching(position1,position2):
  if (real_board[position1]==real_board[position2]): # if positions have same values return true
    return True
  else: #other values
    return False
```

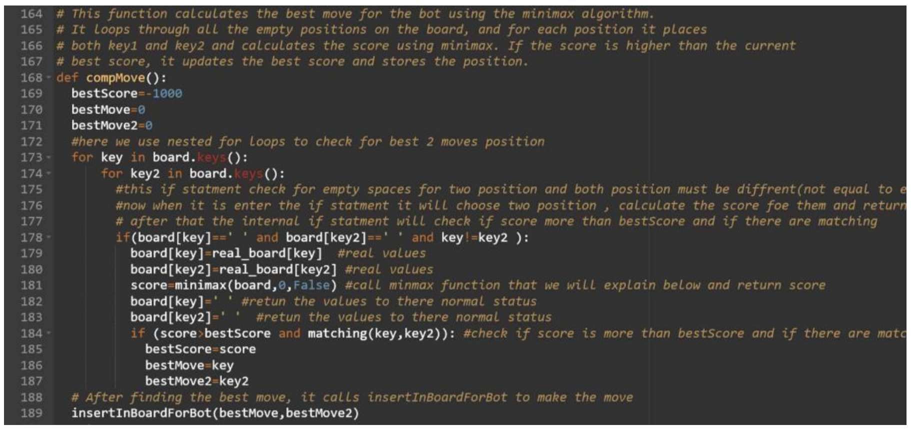
chkDraw() function checks if the game has ended, which happens when all positions on the board are filled, and returns True or False .

The playerMove() function will prompts the player to input two positions, and if they are valid, inserts them into the board using insertInBoardForUser function.



```python

#This function checks if the game has ended, which happens when all positions on the board are filled, and returns True or False
def chkDraw():
    for key in board.keys():
        if (board[key]==' '):
            return False
    return True

 #This function prompts the player to input two positions ,inserts them into the board using insertInBoardForUser function.
def playerMove():
    position1 = int(input('Enter new Position for number 1: ')) #read first position from user take position as int
    position2 = int(input('Enter new Position for number 2: ')) #read second position from user take position as int
    insertInBoardForUser(position1, position2) #call insertInBoardForUser function that we explained above
    return
```

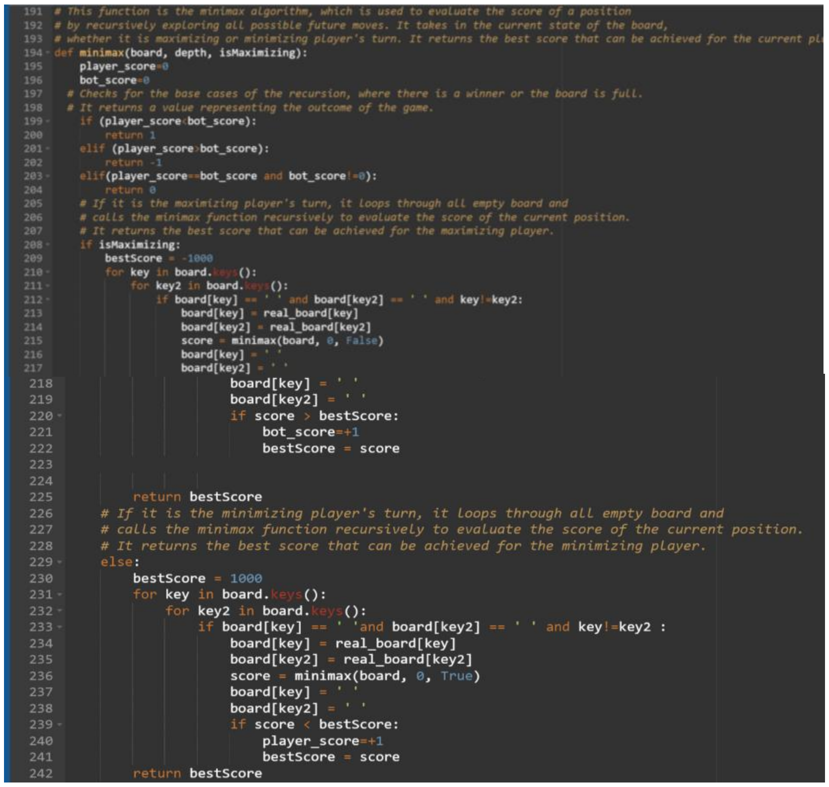
The compMove() function uses minimax algorithm to calculate the best move for the bot.



```python
# This function calculates the best move for the bot using the minimax algorithm.
# It loops through all the empty positions on the board, and for each position it places
# both key1 and key2 and calculates the score using minimax. If the score is higher than the current
# best score, it updates the best score and stores the position.
def compMove():
  bestScore=-1000
  bestMove=0
  bestMove2=0
  #here we use nested for loops to check for best 2 moves position
  for key in board.keys():
      for key2 in board.keys():
        #this if statment check for empty spaces for two position and both position must be diffrent(not equal to e
        #now when it is enter the if statment it will choose two position , calculate the score foe them and return
        # after that the internal if statment will check if score more than bestScore and if there are matching
        if(board[key]==' ' and board[key2]==' ' and key!=key2 ):
          board[key]=real_board[key]  #real values
          board[key2]=real_board[key2] #real values
          score=minimax(board,0,False) #call minmax function that we will explain below and return score
          board[key]=' ' #retun the values to there normal status
          board[key2]=' '  #retun the values to there normal status
          if (score>bestScore and matching(key,key2)): #check if score is more than bestScore and if there are matc
            bestScore=score
            bestMove=key
            bestMove2=key2
  # After finding the best move, it calls insertInBoardForBot to make the move
  insertInBoardForBot(bestMove,bestMove2)
```

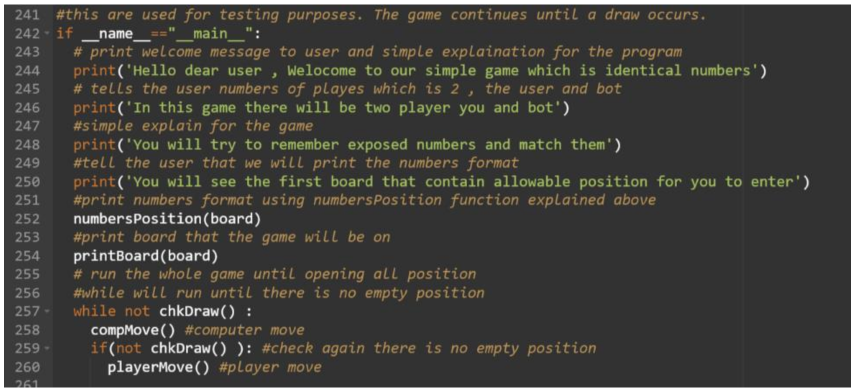
The minimax() function uses minmax algorithm to calculate the best move for the bot return the best move to win the game.

The for loop will keeps looping until it finds 2 identical positions with the highest score to make the move.



```python
# This function is the minimax algorithm, which is used to evaluate the score of a position
# by recursively exploring all possible future moves. It takes in the current state of the board,
# whether it is maximizing or minimizing player's turn. It returns the best score that can be achieved for the current pl
def minimax(board, depth, isMaximizing):
    player_score=0
    bot_score=0
  # Checks for the base cases of the recursion, where there is a winner or the board is full.
  # It returns a value representing the outcome of the game.
    if (player_score<bot_score):
        return 1
    elif (player_score>bot_score):
        return -1
    elif(player_score==bot_score and bot_score!=0):
        return 0
    # If it is the maximizing player's turn, it loops through all empty board and
    # calls the minimax function recursively to evaluate the score of the current position.
    # It returns the best score that can be achieved for the maximizing player.
    if isMaximizing:
        bestScore = -1000
        for key in board.keys():
            for key2 in board.keys():
                if board[key] == ' ' and board[key2] == ' ' and key!=key2:
                    board[key] = real_board[key]
                    board[key2] = real_board[key2]
                    score = minimax(board, 0, False)
                    board[key] = ' '
                    board[key2] = ' '
                    board[key] = ' '
                    board[key2] = ' '
                    if score > bestScore:
                        bot_score=+1
                        bestScore = score


        return bestScore
    # If it is the minimizing player's turn, it loops through all empty board and
    # calls the minimax function recursively to evaluate the score of the current position.
    # It returns the best score that can be achieved for the minimizing player.
    else:
        bestScore = 1000
        for key in board.keys():
            for key2 in board.keys():
                if board[key] == ' 'and board[key2] == ' ' and key!=key2 :
                    board[key] = real_board[key]
                    board[key2] = real_board[key2]
                    score = minimax(board, 0, True)
                    board[key] = ' '
                    board[key2] = ' '
                    if score < bestScore:
                        player_score=+1
                        bestScore = score
        return bestScore
```

This is the main function to start the game execution by calling the compMove function to let the bot do its first move.



```python
#this are used for testing purposes. The game continues until a draw occurs.
if __name__=="__main__":
  # print welcome message to user and simple explaination for the program
  print('Hello dear user , Welocome to our simple game which is identical numbers')
  # tells the user numbers of playes which is 2 , the user and bot
  print('In this game there will be two player you and bot')
  #simple explain for the game
  print('You will try to remember exposed numbers and match them')
  #tell the user that we will print the numbers format
  print('You will see the first board that contain allowable position for you to enter')
  #print numbers format using numbersPosition function explained above
  numbersPosition(board)
  #print board that the game will be on
  printBoard(board)
  # run the whole game until opening all position
  #while will run until there is no empty position
  while not chkDraw() :
    compMove() #computer move
    if(not chkDraw() ): #check again there is no empty position
      playerMove() #player move
```