

```

1 import random
2
3
4 # The starter function, lets the user decide how to play the game
5 def gamePick():
6     global gamePick
7     print('\tTic-Tac-Toe')
8
9     options = ['2 Players local play', 'Unbeatable Computer']
10    for i in range(2):
11        print(f"{i + 1}) {options[i]}")
12
13    while True:
14        # Validating the input
15        gamePick = str(input('Please pick your game type: '))
16
17        # These conditions are seperated to check first if the input not a number
18        # and only after that check if its withing the range of options
19        if not gamePick.isdigit():
20            print('✗: Wrong input for gamePick, try again')
21        elif int(gamePick) > len(options) or int(gamePick) < 1:
22            print('✗: Wrong input for gamePick, try again')
23        else:
24            break
25
26    gamePick = int(gamePick) - 1
27    if gamePick == 0:
28        localPlay()
29    elif gamePick == 1:
30        vsComp()
31
32
33 # Switching the currently playing player
34 def switchPlayer():
35     if player['sign'] == 'X':
36         player['sign'] = 'O'
37
38     # Checks if you're playing against ai
39     if(gamePick == 1):
40         player['name'] = 'Computer'
41     else:
42         player['name'] = secPlayerName
43 else:
44     player['sign'] = 'X'
45     player['name'] = firstPlayerName
46
47
48 # Reset the board, get the players names and flip a coin to see who's playing first
49 def reset(isFirstTime=False):
50     global board, player, firstPlayerName, secPlayerName, score
51     if isFirstTime:
52         firstPlayerName = str(input('Please enter player1 name: '))
53
54     # gamePick 0 = [2 players local play] => need a second players name
55     if gamePick == 0:
56         secPlayerName = str(input('Please enter player2 name: '))
57         score = {
58             firstPlayerName: 0,
59             secPlayerName: 0

```

```

60     }
61     else:
62         score = {
63             firstPlayerName: 0
64         }
65
66         player = {
67             'name': firstPlayerName,
68             'sign': 'X',
69         }
70     board = [['_', '_', '_'],
71              ['_', '_', '_'],
72              ['_', '_', '_']]
73
74     coinFlipPick = str(input('Please enter a coin flip pick(0/1): '))
75     randomCoinFlip = str(random.randint(0, 1))
76     print(f"Flipped the coin: {randomCoinFlip}")
77     if coinFlipPick != randomCoinFlip:
78         switchPlayer()
79
80     if not isFirstTime:
81         if gamePick == 0:
82             localPlay(isFirstTime)
83         elif gamePick == 1:
84             vsComp(isFirstTime)
85
86
87 def printBoard():
88     print('    0    1    2')
89     col = 0
90     for row in range(3):
91         print(f"{col} {board[row]}")
92         col += 1
93
94
95 # Returns the winner sign if there's a winner, if not it will return None to continue
96 # playing
97 def getWinner():
98     # Row win
99     for row in range(3):
100         if '_' not in board[row]:
101             if len(set(board[row])) == 1:
102                 return board[row][0]
103
104     # Column win
105     for row in range(3):
106         column = []
107         for col in range(3):
108             if board[col][row] != '_':
109                 column.append(board[col][row])
110             if len(set(column)) == 1 and len(column) == 3:
111                 return column[0]
112
113     # Crosses win
114     cross1 = []
115     cross2 = []
116     for i in range(3):
117         if board[i][i] != '_':
118             cross1.append(board[i][i])
119         if len(set(cross1)) == 1 and len(cross1) == 3:

```

```

119         return board[i][i]
120     if board[i][2 - i] != '_':
121         cross2.append(board[i][2 - i])
122         if len(set(cross2)) == 1 and len(cross2) == 3:
123             return board[i][2 - i]
124
125     # No win, board is not full and its checking this condition
126     for row in range(0, 3):
127         for col in range(0, 3):
128             if (board[row][col] == '_'):
129                 return None
130
131     # Draw
132     return '_'
133
134
135 def isValidFormatInput():
136     while True:
137         move = str(
138             input('Please enter the indexes for your move(x,y): '))
139         try:
140             [x, y] = [int(x) for x in move.split(',')]
141             return [x, y]
142         except Exception:
143             print('✗: Wrong format, try again using the correct format: x,y')
144
145
146 def isValidMove():
147     while True:
148         [x, y] = isValidFormatInput()
149
150         # Check if one of the indexes is out of range
151         # or longer than the accepted format
152         if (
153             x > 2 or x < 0 or
154             y > 2 or y < 0 or
155             board[x][y] != '_'
156         ):
157             print('✗: Not a valid move, enter the indexes again please: ')
158             # return False
159         else:
160             return [x, y]
161
162
163 ...
164
165     One of two functions,
166     this function will try to minimize the opponents score.
167     Using recursion in order to test every board possibillity, the function check
168     if the possibility tree is making the opponent lose.
169     it will return a value by this terms:
170         1: win
171         0: draw
172         -1: loss
173
174 ...
175
176 def minCompMove():
177
178     # Initialize the outcome of the move with the lowest possible value
179     outcome = 2

```

```

179
180 x = None
181 y = None
182 result = getWinner()
183
184 # X wins = computer loss
185 if result == 'X':
186     return (-1, 0, 0)
187
188 # O wins = computer wins
189 elif result == 'O':
190     return (1, 0, 0)
191
192 # _ = draw
193 elif result == '_':
194     return (0, 0, 0)
195
196 # Find the best outcome for all unused cells
197 for row in range(3):
198     for col in range(3):
199         if board[row][col] == '_':
200             # Creating a possibility recourse tree
201             board[row][col] = 'X'
202             [m, maxRow, maxCol] = compMove()
203             if m < outcome:
204                 outcome = m
205                 x = row
206                 y = col
207             # Revert Changes
208             board[row][col] = '_'
209
210 return [outcome, x, y]
211
212 '''
213 One of two functions,
214 This function will try to maximize the computers score.
215 Using recursion in order to test every board possibillity, the function check
216 if the possibility tree is getting us to win with the least moves.
217 it will return a value by this terms:
218     1: win
219     0: draw
220     -1: loss
221 '''
222
223
224
225 def compMove():
226
227     # Initialize the outcome of the move with the lowest possible value
228     outcome = -2
229
230     x = None
231     y = None
232     result = getWinner()
233
234     if result == 'X':
235         return (-1, 0, 0)
236     elif result == 'O':
237         return (1, 0, 0)
238     elif result == '_':

```

```

239         return (0, 0, 0)
240
241     for row in range(3):
242         for col in range(3):
243             if board[row][col] == '_':
244                 # Creating a possibility recourse tree
245                 board[row][col] = 'O'
246                 [m, minRow, minCol] = minCompMove()
247                 if m > outcome:
248                     outcome = m
249                     x = row
250                     y = col
251                 # Revert Changes
252                 board[row][col] = '_'
253     return [outcome, x, y]
254
255
256 # Checks if the board is empty
257 def isFirstMove():
258     for row in range(0, 3):
259         for col in range(0, 3):
260             if (board[row][col] != '_'):
261                 return False
262     return True
263
264
265 def printScores():
266     sortedScores = sorted(
267         score.items(), key=lambda scoreData: scoreData[1], reverse=True)
268     for index, scoreTuple in enumerate(sortedScores):
269         print(f"{index + 1}) {scoreTuple[1]}: {scoreTuple[0]}")
270
271
272 def updateScore(winnerSign):
273     if winnerSign == 'draw':
274         if not player['name'] == 'Computer':
275             score[player['name']] += 1
276
277     # Check if theres only one player(happens when playing vs computer)
278     if len(score.keys()) > 1:
279         switchPlayer()
280         if not player['name'] == 'Computer':
281             score[player['name']] += 1
282     else:
283
284         # A short condition to check if the winner is the currently active player.
285         if player['sign'] != winnerSign[0]:
286             switchPlayer()
287         if not player['name'] == 'Computer':
288             score[player['name']] += 2
289
290
291 def humanMove(playerSign):
292     [x, y] = isValidMove()
293     board[x][y] = playerSign
294     switchPlayer()
295
296
297 def restart():
298     # Optional restart after the game has finished

```

```

299     print('Type showScores to see the score board')
300     answer = str(
301         input('Would you like to restart(Y/N/showScores): ').lower()
302     if answer == 'showscores':
303         printScores()
304         restart()
305     elif answer == 'y':
306         reset()
307     else:
308         print('Thank you for playing!')
309         return False
310
311 def vsComp(isFirstTime=True):
312     if isFirstTime:
313         reset(True)
314
315     while True:
316         printBoard()
317         winner = getWinner()
318
319         if winner != None:
320             if winner == 'X':
321                 print('X Wins.')
322                 updateScore('X')
323             elif winner == 'O':
324                 print('O Wins.')
325                 updateScore('O')
326             elif winner == '_':
327                 print('Draw!')
328                 updateScore('draw')
329
330             # Checking if the user wants to end the game or not
331             if not restart():
332                 return
333
334     print(player['name'] + '\s turn as ' + player['sign'])
335
336     # User's turn
337     if player['sign'] == 'X':
338         humanMove('X')
339
340     # Computer's turn
341     else:
342         if isFirstMove():
343             board[random.randint(0, 2)][random.randint(0, 2)] = 'O'
344         else:
345             [status, x, y] = compMove()
346             board[x][y] = 'O'
347             switchPlayer()
348
349
350 def localPlay(isFirstTime=True):
351     if isFirstTime:
352         reset(True)
353
354     while True:
355         printBoard()
356         winner = getWinner()
357
358

```

```
359     if winner != None:
360         if winner == 'X':
361             print('X Wins.')
362             updateScore('X')
363         elif winner == 'O':
364             print('O Wins.')
365             updateScore('O')
366         elif winner == '_':
367             print('Draw!')
368             updateScore('draw')
369
370         # Checking if the user wants to end the game or not
371         if not restart():
372             return
373
374     print(player['name'] + '\s turn as ' + player['sign'])
375
376     humanMove(player['sign'])
377
378
379 gamePick()
380
```