CS50p Final Project

Programmer Profile

I'm **Saleh Miri** from Tehran, Iran.
Linkedin.com/in/salehmiri

Project Name

Backgammon GAME for two players.

Programming Language

Python.

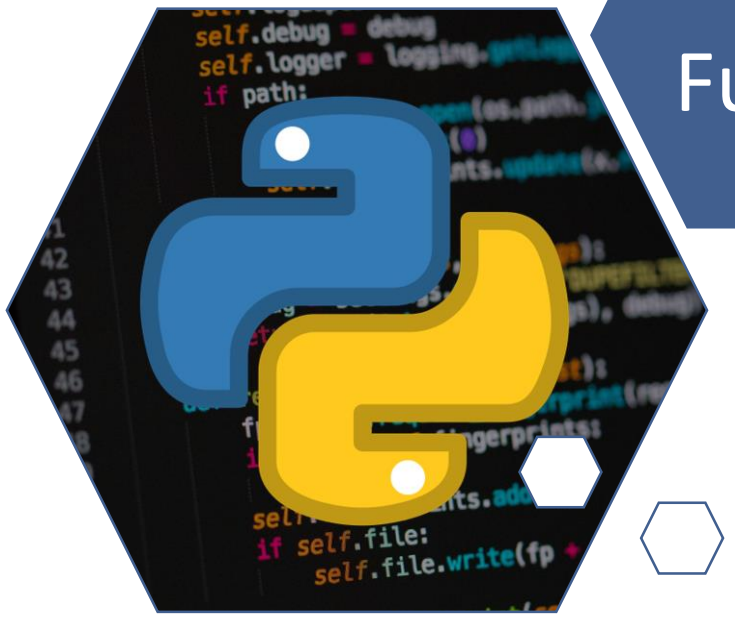
Project Description

This game is allows two players to play the game. The game is set up on a board with 24 triangles known as points. Each player starts with 15 checkers, and the objective of the game is to get all of one's pieces off the board as quickly as possible. The legal moves are determined by rolling dice at the start of each player's turn.

Project Structure

The program consists of a main function and 9 additional functions namely: play_game, initialize_board, print_board, get_move, roll_dice, can_move, make_move, check_winner and switch_player.

Main Function



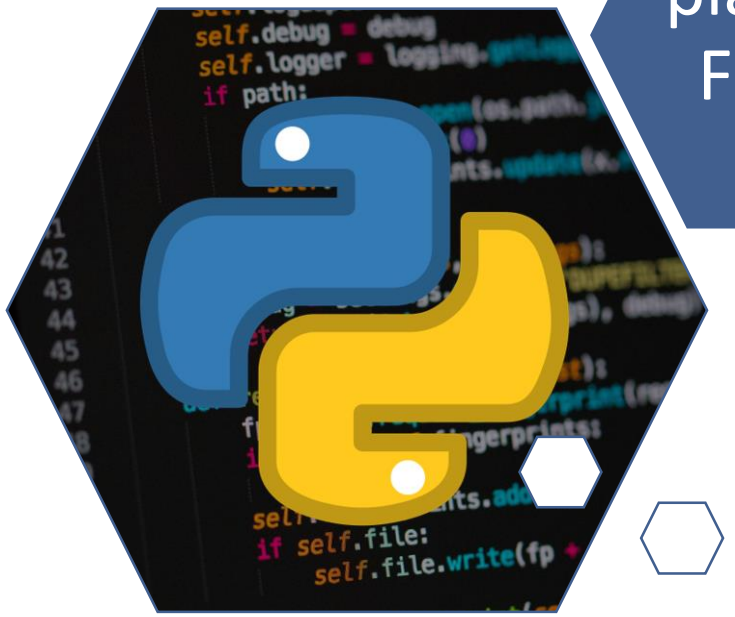
This function is the entry point of the program and prints a welcome message.

It then calls the play_game() function to start the game.

```
2
3   def main():
4       print("Welcome to Backgammon!")
5       play_game()
6
```

Contact Programmer
Linkedin.com/in/**SalehMiri**

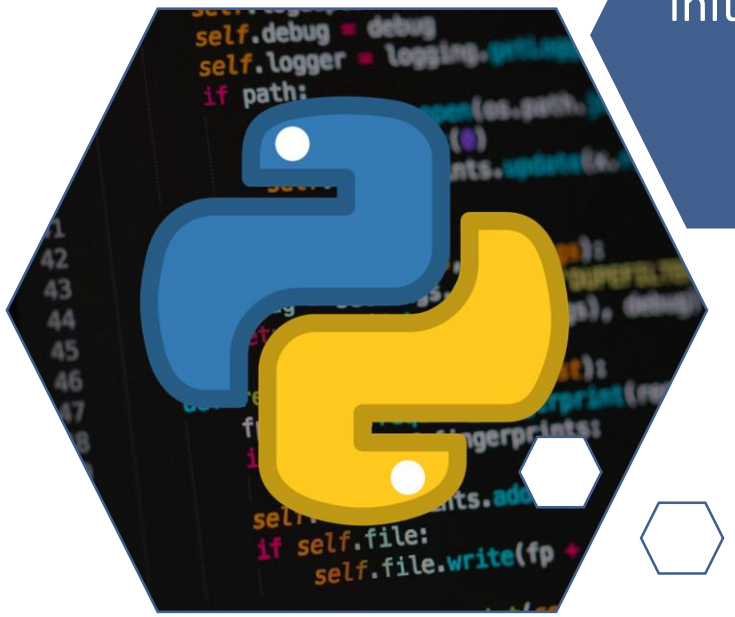
play_game Function



This function initializes the board, sets the current player to "X", and starts a loop that continues until the game is over. In each iteration of the loop, it prints the current state of the board, gets the move for the current player using the get_move() function, makes the move using the make_move() function, and checks if the game is over using the check_winner() function. If the game is over, it prints the winner and sets the game_over flag to True. Otherwise, it switches the current player using the switch_player() function.

```
7   def play_game():
8       board = initialize_board()
9       current_player = "X"
10      game_over = False
11
12      while not game_over:
13          print_board(board)
14          move = get_move(current_player, board)
15          make_move(board, move, current_player)
16          if check_winner(board, current_player):
17              print("Player", current_player, "wins!")
18              game_over = True
19          else:
20              current_player = switch_player(current_player)
```

Contact Programmer
Linkedin.com/in/**SalehMiri**

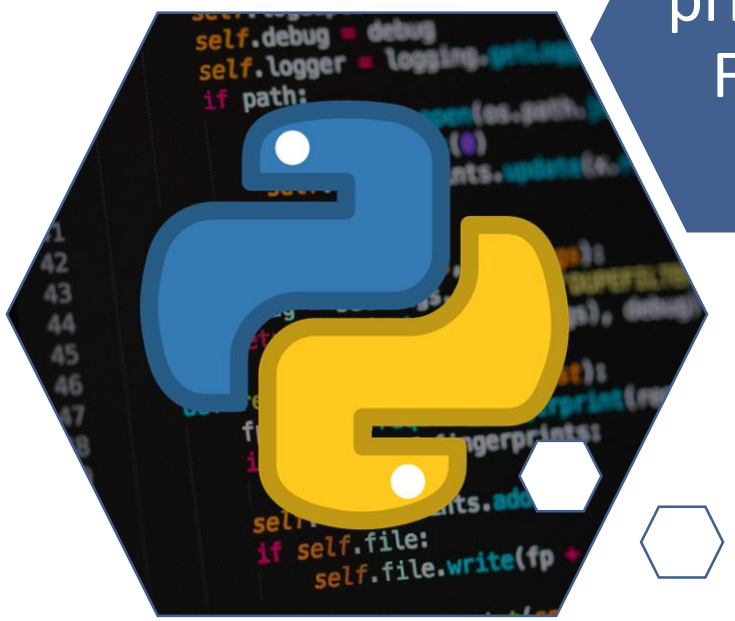


initialize_board Function

This function returns a dictionary that represents the initial state of the board. Each key in the dictionary represents a point on the board, and the value is a list of checkers on that point. The first element of the list represents the color of the checkers on that point.

```
22  def initialize_board():
23      return {
24          1: ["X", "X", "X", "X", "X"],
25          6: ["O", "O", "O", "O", "O"],
26          8: ["O", "O", "O", "O", "O"],
27          12: ["X", "X", "X", "X", "X"],
28          13: ["O", "O", "O", "O", "O"],
29          17: ["X", "X", "X", "X", "X"],
30          19: ["X", "X", "X", "X", "X"],
31          24: ["O", "O", "O", "O", "O"]
32      }
```

Contact Programmer
Linkedin.com/in/**SalehMiri**

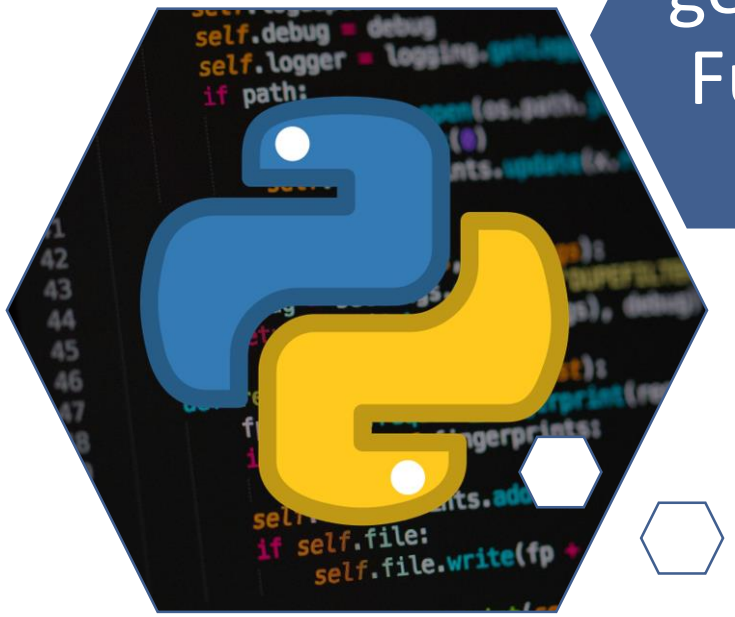


print_board Function

This function takes a board dictionary as input and prints the current state of the board to the console.

```
34  def print_board(board):
35      for i in range(24, 0, -1):
36          if i in board:
37              print("|".join(board[i]), end=" ")
38          else:
39              print("| |", end=" ")
40          if i % 6 == 1:
41              print()
```

Contact Programmer
Linkedin.com/in/**SalehMiri**



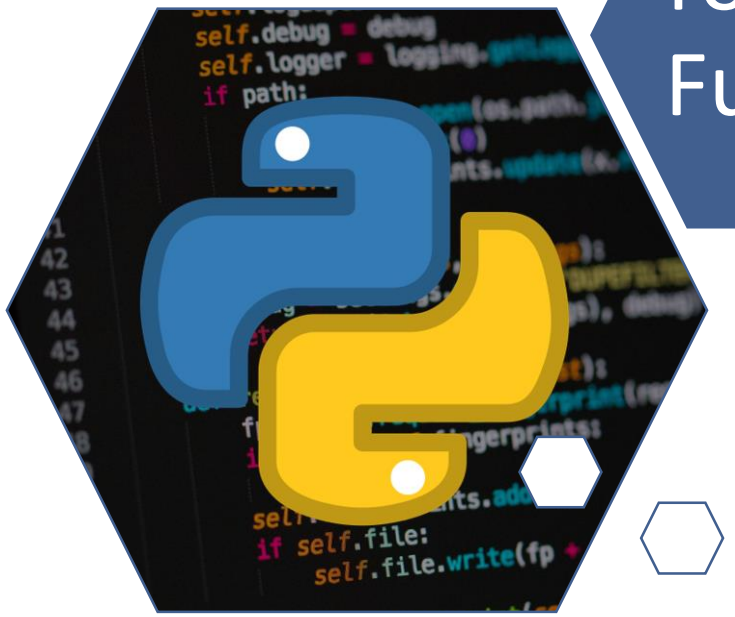
get_move Function

This function takes the current player and the board dictionary as input and returns a list of moves for the player. It rolls two dice using the roll_dice() function and prompts the player to enter a move for each roll. It checks if the move is valid using the can_move() function and returns a list of valid moves.

```
43  def get_move(player, board):
44      roll1, roll2 = roll_dice()
45      print("Player", player, "rolls", roll1, "and", roll2)
46      moves = []
47      for roll in [roll1, roll2]:
48          if can_move(player, board, roll):
49              move = input("Enter move for roll " + str(roll) + ": ")
50              moves.append((move, roll))
51          else:
52              moves.append(None)
53      return moves
```

Contact Programmer
Linkedin.com/in/**SalehMiri**

roll_dice Function

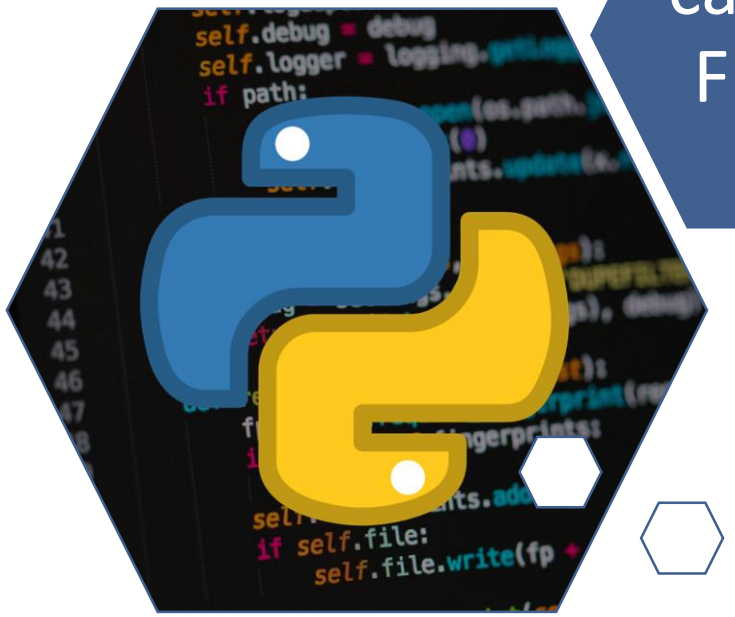


This function returns two random integers between 1 and 6, which represent the values of two dice.

```
55  def roll_dice():
56      return random.randint(1, 6), random.randint(1, 6)
```

Contact Programmer
Linkedin.com/in/**SalehMiri**

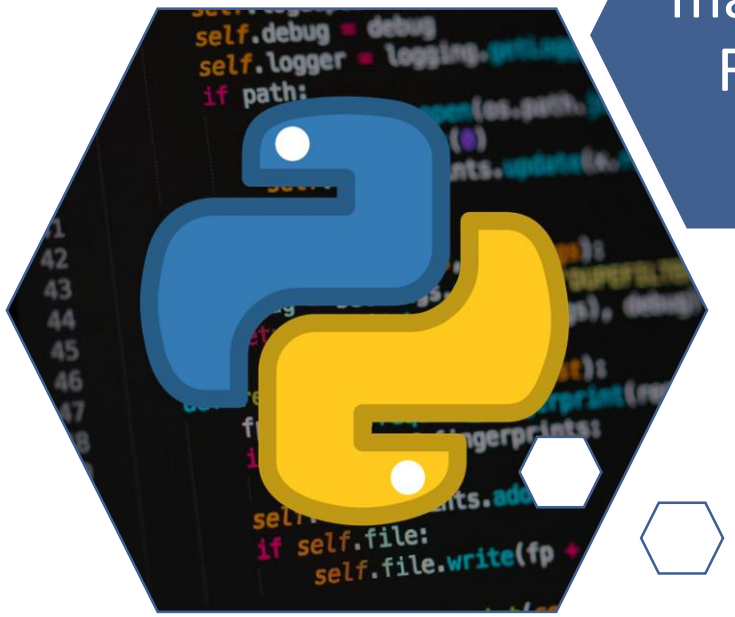
can_move Function



This function takes the current player, the board dictionary, and the value of a roll as input and returns True if the player can make a move with the given roll, and False otherwise.

```
58  def can_move(player, board, roll):
59      for i in range(1, 25):
60          if i in board and board[i][0] == player:
61              if i + roll in board:
62                  if len(board[i+roll]) <= 5 or board[i+roll][0] == player:
63                      return True
64              else:
65                  return True
66      return False
```

Contact Programmer
Linkedin.com/in/**SalehMiri**

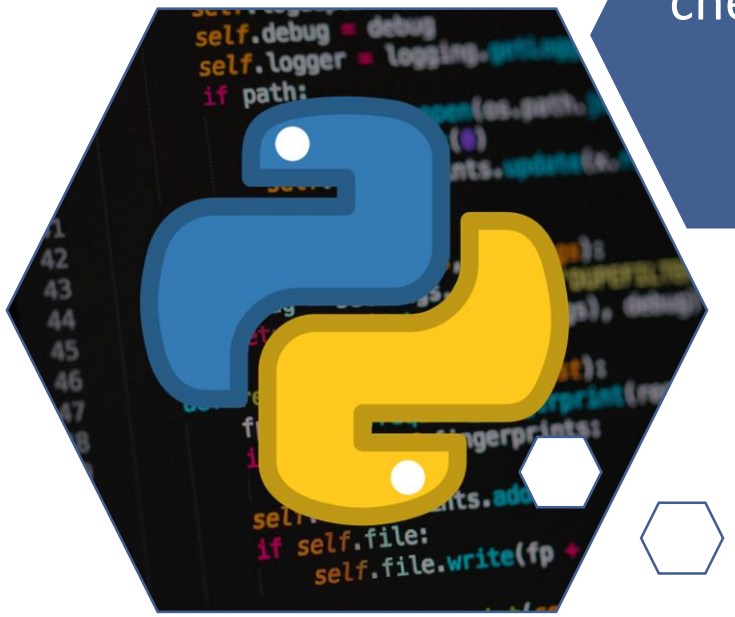


make_move Function

This function takes the board dictionary, a list of moves, and the current player as input and updates the board according to the moves made by the player.

```
68  def make_move(board, moves, player):
69      for move in moves:
70          if move:
71              src, dest = move[0].split("-")
72              src, dest = int(src), int(dest)
73              if src in board and board[src][0] == player:
74                  if dest in board:
75                      if len(board[dest]) <= 5 or board[dest][0] == player:
76                          board[src].pop(0)
77                          board[dest].insert(0, player)
78                  else:
79                      board[src].pop(0)
80                      board[dest] = [player]
```

Contact Programmer
Linkedin.com/in/**SalehMiri**

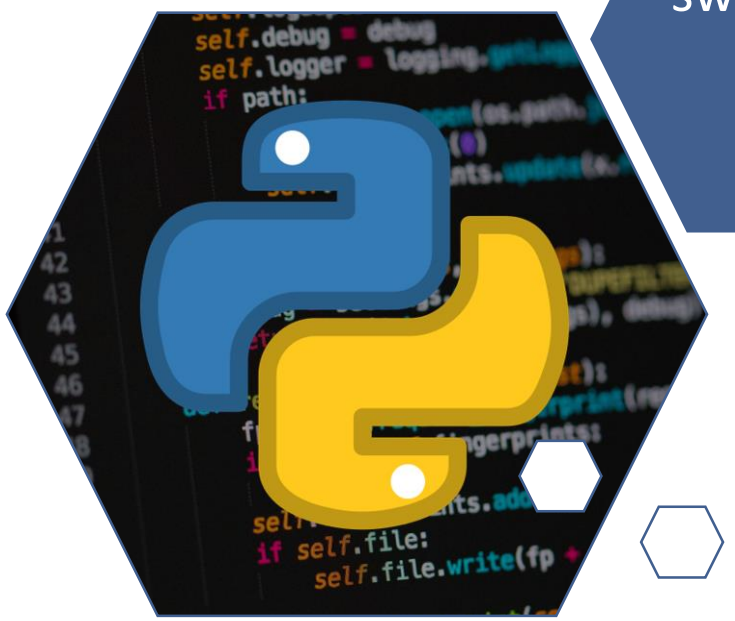


check_winner Function

This function takes the board dictionary and the current player as input and returns True if the player has won the game, and False otherwise.

```
82    def check_winner(board, player):
83        return all(cell == player for cell in board[1][:5]) or all(cell == player for cell in board[24][:5])
```

Contact Programmer
Linkedin.com/in/**SalehMiri**



switch_player Function

This function takes the current player as input and returns the other player.

```
85  def switch_player(player):
86      return "O" if player == "X" else "X"
```

Contact Programmer
Linkedin.com/in/**SalehMiri**