“Tic Tac Toe”

Invent Your Own Computer Games with Python

Introduction

- "Tic Tac Toe"
  - The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row wins the game.
  - Sample Run
  - Source Code
- Designing the Program
- Code Explanation
- Things Covered In This Chapter

How to play tic tac toe

Sample Run of Tic Tac Toe

```
Welcome to Tic Tac Toe!
Do you want to be X or O?
X
The computer will go first.
   |   |
 O |   |
   |   |
-----------
   |   |
   |   |
   |   |
-----------
   |   |
   |   |
   |   |


What is your next move? (1-9)
4
   |   |
 O |   | O
   |   |
-----------
   |   |
 X |   |
   |   |
-----------
   |   |
 O |   | X
   |   |
```

```
What is your next move? (1-9)
3
   |   |
 O |   |
   |   |
-----------
   |   |
   |   |
   |   |
-----------
   |   |
 O |   | X
   |   |


What is your next move? (1-9)
5
   |   |
 O | O | O
   |   |
-----------
   |   |
 X | X |
   |   |
-----------
   |   |
 O |   | X
   |   |
The computer has beaten you! You lose.
Do you want to play again? (yes or no)
no
```

Source Code of Tic Tac Toe (1/7)

```python
# Tic Tac Toe

import random

def drawBoard(board):
    # This function prints out the board that it was passed.

    # "board" is a list of 10 strings representing the board (ignore index 0)
    print(board[7] + '|' + board[8] + '|' + board[9])
    print('-+-+-')
    print(board[4] + '|' + board[5] + '|' + board[6])
    print('-+-+-')
    print(board[1] + '|' + board[2] + '|' + board[3])

def inputPlayerLetter():
    # Lets the player type which letter they want to be.
    # Returns a list with the player's letter as the first item, and the computer's letter as the second.
    letter = ''
    while not (letter == 'X' or letter == 'O'):
        print('Do you want to be X or O?')
        letter = input().upper()

    # the first element in the list is the player's letter, the second is the computer's letter.
    if letter == 'X':
        return ['X', 'O']
    else:
        return ['O', 'X']
```

Source Code of Tic Tac Toe (2/7)

```
def whoGoesFirst():
    # Randomly choose the player who goes first.
    if random.randint(0, 1) == 0:
        return 'computer'
    else:
        return 'player'

def makeMove(board, letter, move):
    board[move] = letter

def isWinner(bo, le):
    # Given a board and a player's letter, this function returns True if that player has won.
    # We use bo instead of board and le instead of letter so we don't have to type as much.
    return ((bo[7] == le and bo[8] == le and bo[9] == le) or # across the top
    (bo[4] == le and bo[5] == le and bo[6] == le) or # across the middle
    (bo[1] == le and bo[2] == le and bo[3] == le) or # across the bottom
    (bo[7] == le and bo[4] == le and bo[1] == le) or # down the left side
    (bo[8] == le and bo[5] == le and bo[2] == le) or # down the middle
    (bo[9] == le and bo[6] == le and bo[3] == le) or # down the right side
    (bo[7] == le and bo[5] == le and bo[3] == le) or # diagonal
    (bo[9] == le and bo[5] == le and bo[1] == le)) # diagonal

def getBoardCopy(board):
    # Make a copy of the board list and return it.
    boardCopy = []
    for i in board:
        boardCopy.append(i)
    return boardCopy
```

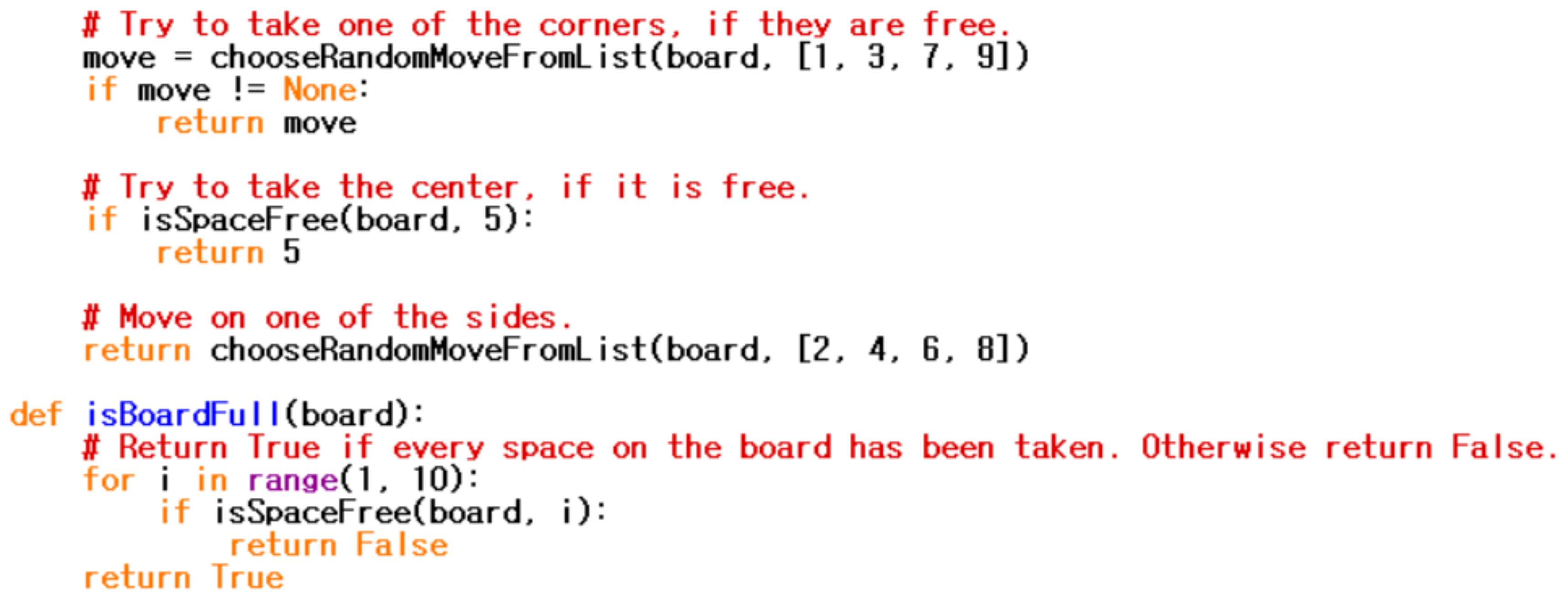
Source Code of Tic Tac Toe (3/7)

```
def isSpaceFree(board, move):
    # Return true if the passed move is free on the passed board.
    return board[move] == ' '

def getPlayerMove(board):
    # Let the player type in their move.
    move = ' '
    while move not in '1 2 3 4 5 6 7 8 9'.split() or not isSpaceFree(board, int(move)):
        print('What is your next move? (1-9)')
        move = input()
    return int(move)

def chooseRandomMoveFromList(board, movesList):
    # Returns a valid move from the passed list on the passed board.
    # Returns None if there is no valid move.
    possibleMoves = []
    for i in movesList:
        if isSpaceFree(board, i):
            possibleMoves.append(i)

    if len(possibleMoves) != 0:
        return random.choice(possibleMoves)
    else:
        return None
```

Source Code of Tic Tac Toe (4/7)

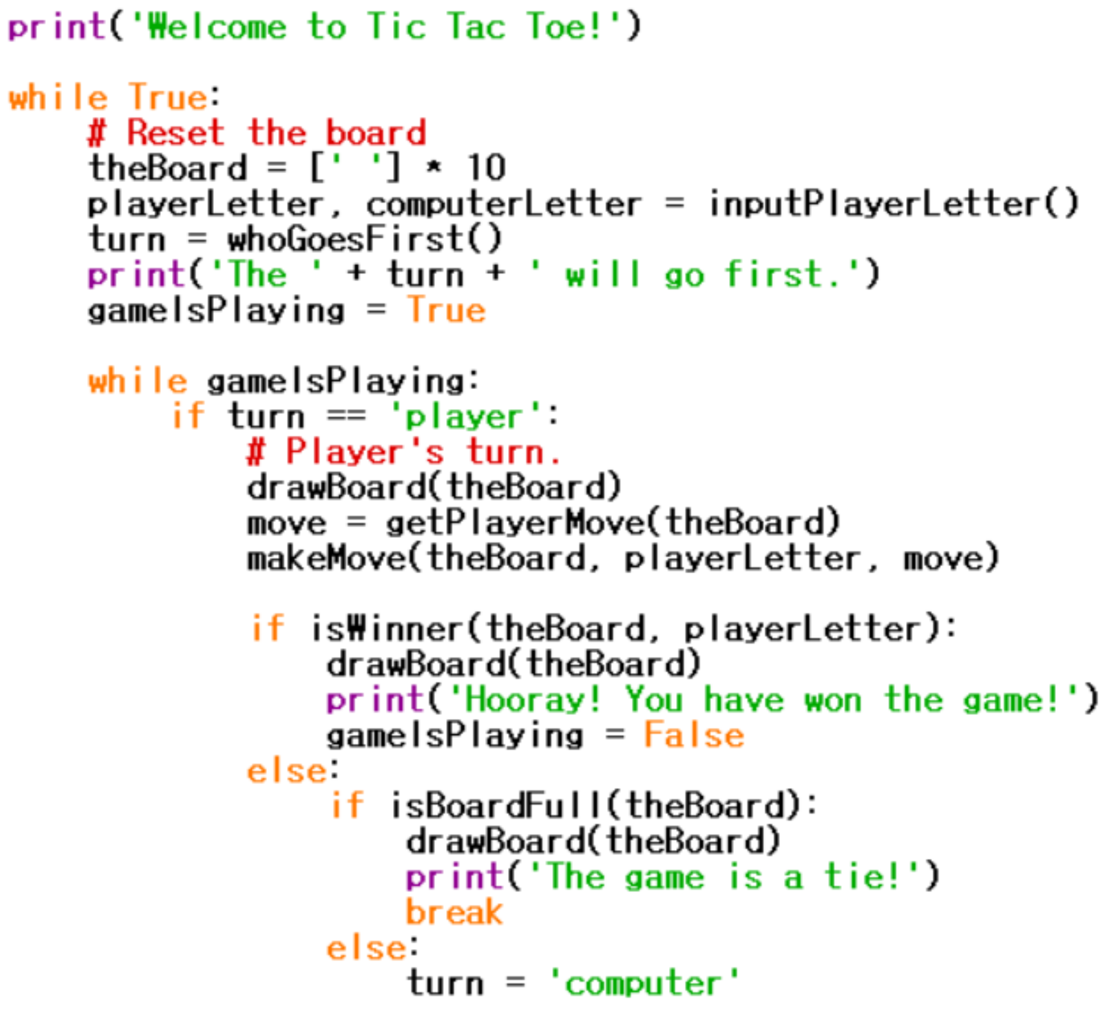
```
def getComputerMove(board, computerLetter):
    # Given a board and the computer's letter, determine where to move and return that move.
    if computerLetter == 'X':
        playerLetter = 'O'
    else:
        playerLetter = 'X'

    # Here is our algorithm for our Tic Tac Toe AI:
    # First, check if we can win in the next move
    for i in range(1, 10):
        boardCopy = getBoardCopy(board)
        if isSpaceFree(boardCopy, i):
            makeMove(boardCopy, computerLetter, i)
            if isWinner(boardCopy, computerLetter):
                return i

    # Check if the player could win on his next move, and block them.
    for i in range(1, 10):
        boardCopy = getBoardCopy(board)
        if isSpaceFree(boardCopy, i):
            makeMove(boardCopy, playerLetter, i)
            if isWinner(boardCopy, playerLetter):
                return i
```

Source Code of Tic Tac Toe (5/7)



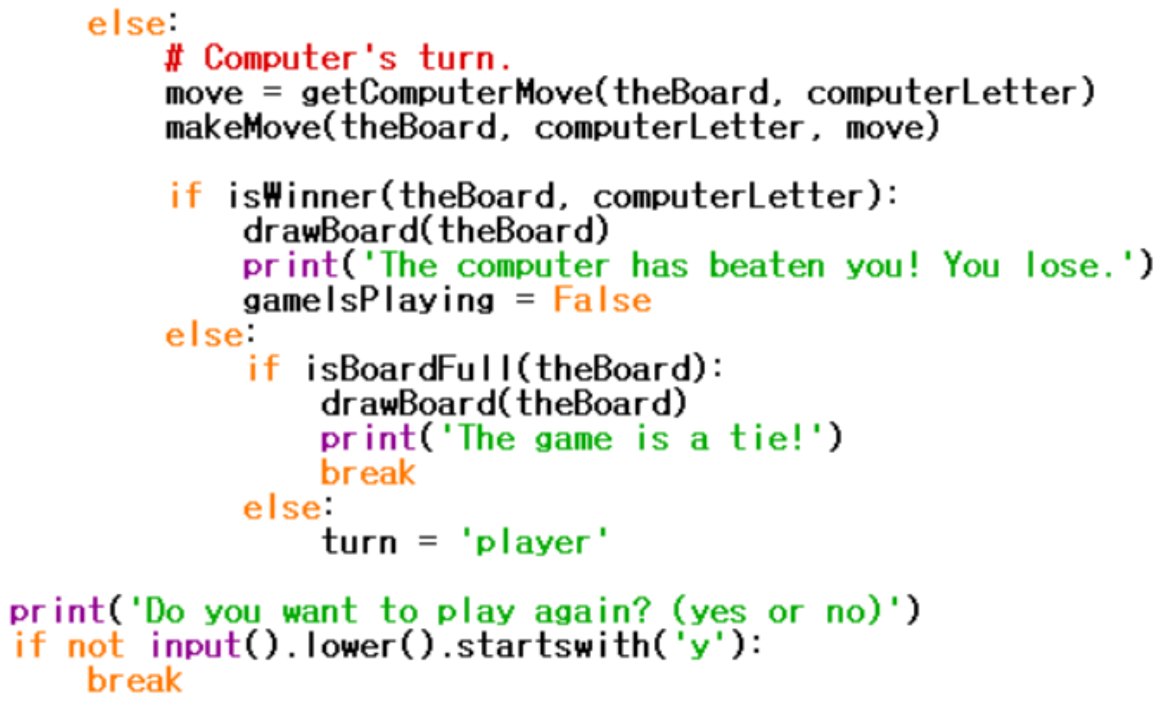
```
    # Try to take one of the corners, if they are free.
    move = chooseRandomMoveFromList(board, [1, 3, 7, 9])
    if move != None:
        return move

    # Try to take the center, if it is free.
    if isSpaceFree(board, 5):
        return 5

    # Move on one of the sides.
    return chooseRandomMoveFromList(board, [2, 4, 6, 8])

def isBoardFull(board):
    # Return True if every space on the board has been taken. Otherwise return False.
    for i in range(1, 10):
        if isSpaceFree(board, i):
            return False
    return True
```

Source Code of Tic Tac Toe (6/7)



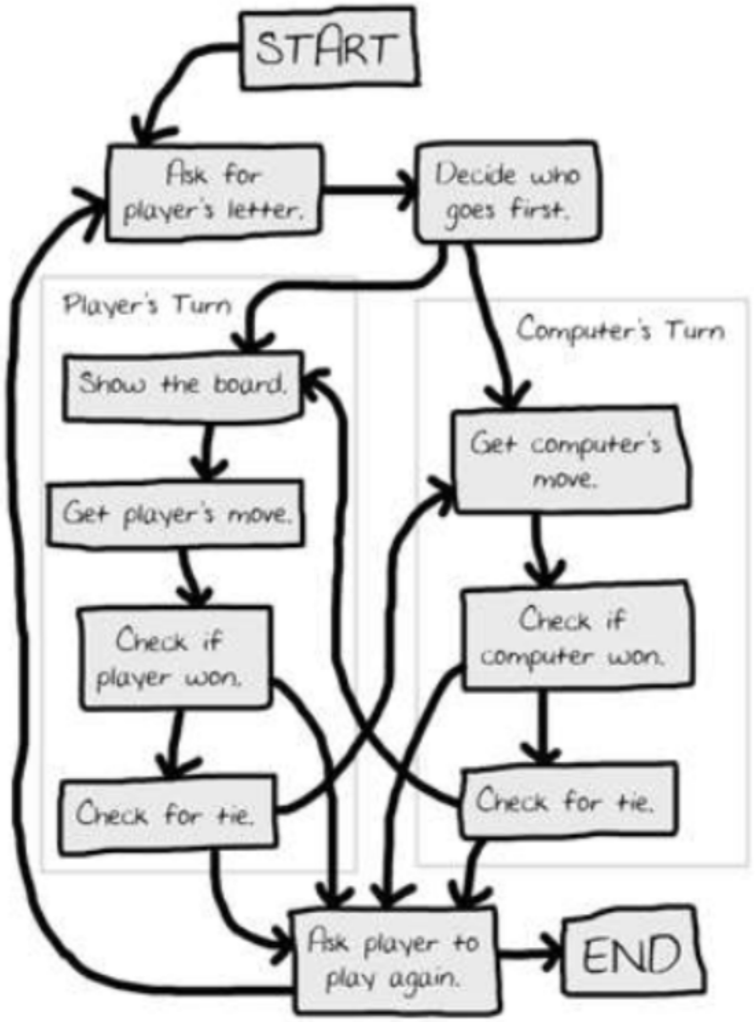
```
print('Welcome to Tic Tac Toe!')

while True:
    # Reset the board
    theBoard = [' '] * 10
    playerLetter, computerLetter = inputPlayerLetter()
    turn = whoGoesFirst()
    print('The ' + turn + ' will go first.')
    gameIsPlaying = True

    while gameIsPlaying:
        if turn == 'player':
            # Player's turn.
            drawBoard(theBoard)
            move = getPlayerMove(theBoard)
            makeMove(theBoard, playerLetter, move)

            if isWinner(theBoard, playerLetter):
                drawBoard(theBoard)
                print('Hooray! You have won the game!')
                gameIsPlaying = False
            else:
                if isBoardFull(theBoard):
                    drawBoard(theBoard)
                    print('The game is a tie!')
                    break
                else:
                    turn = 'computer'
```

Source Code of Tic Tac Toe (7/7)



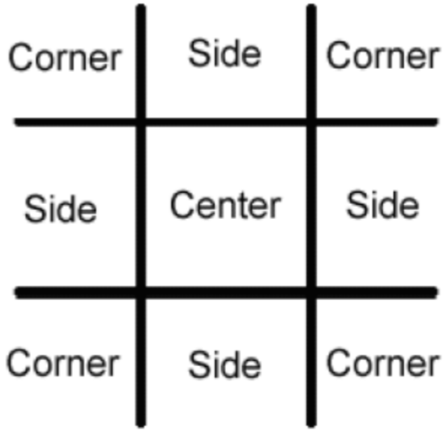
```
        else:
            # Computer's turn.
            move = getComputerMove(theBoard, computerLetter)
            makeMove(theBoard, computerLetter, move)

            if isWinner(theBoard, computerLetter):
                drawBoard(theBoard)
                print('The computer has beaten you! You lose.')
                gameIsPlaying = False
            else:
                if isBoardFull(theBoard):
                    drawBoard(theBoard)
                    print('The game is a tie!')
                    break
                else:
                    turn = 'player'

    print('Do you want to play again? (yes or no)')
    if not input().lower().startswith('y'):
        break
```

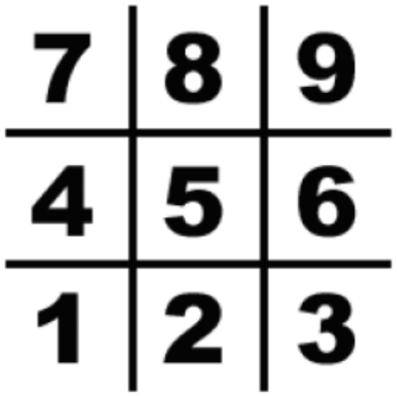
Flow Chart for Tic Tac Toe

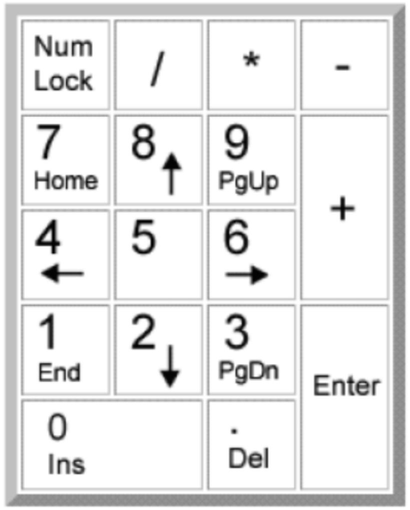


Designing the Program

- The board is numbered like the keyboard's number pad



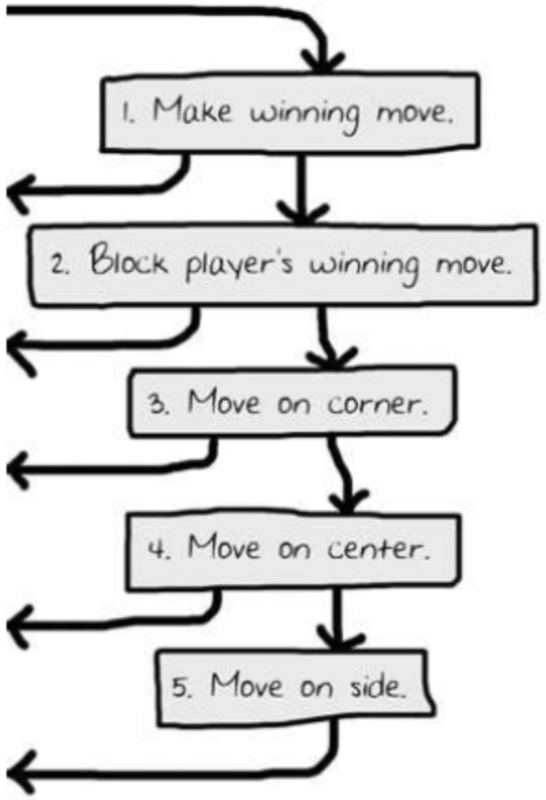




Designing the Program

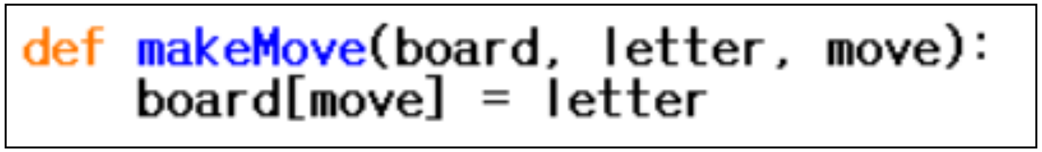
- **Game AI**
  - Algorithm: a finite series of instructions to compute a result
- **Algorithm**

1. See if there's a winning move for the computer
2. If there's a player's winning move, move there to block the player
3. Take any of the corner spaces (1, 3, 7, 9), if available
4. Take the center, if free
5. Take any of the side pieces (2, 4, 6, 8), if available. Otherwise, no more spaces left.



References

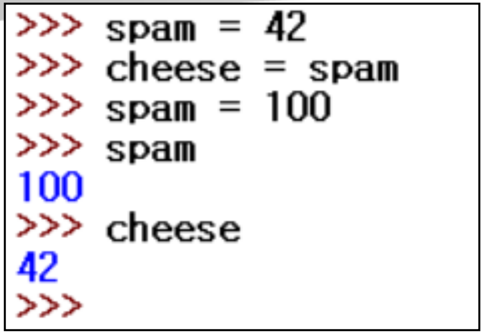
- **Function with a variable of global scope**



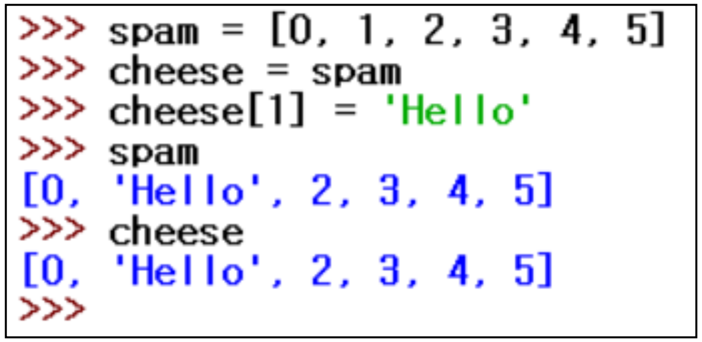
```python
def makeMove(board, letter, move):
    board[move] = letter
```

- Will any changes to the `board` parameter be forgotten, when the functions ends?
- But it isn’t the case where a list is passed to functions.
- You are actually passing **a reference of the list** and not the list itself.

References



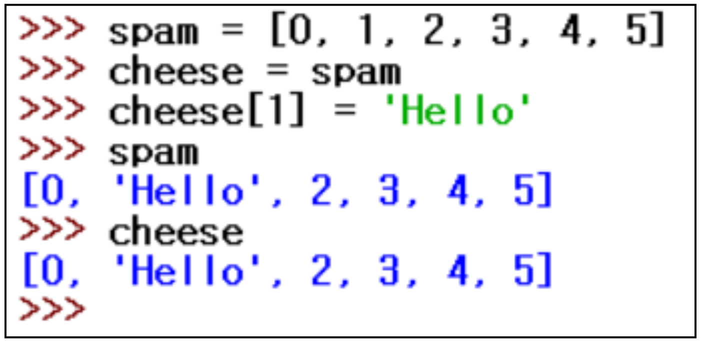
```
>>> spam = 42
>>> cheese = spam
>>> spam = 100
>>> spam
100
>>> cheese
42
>>>
```



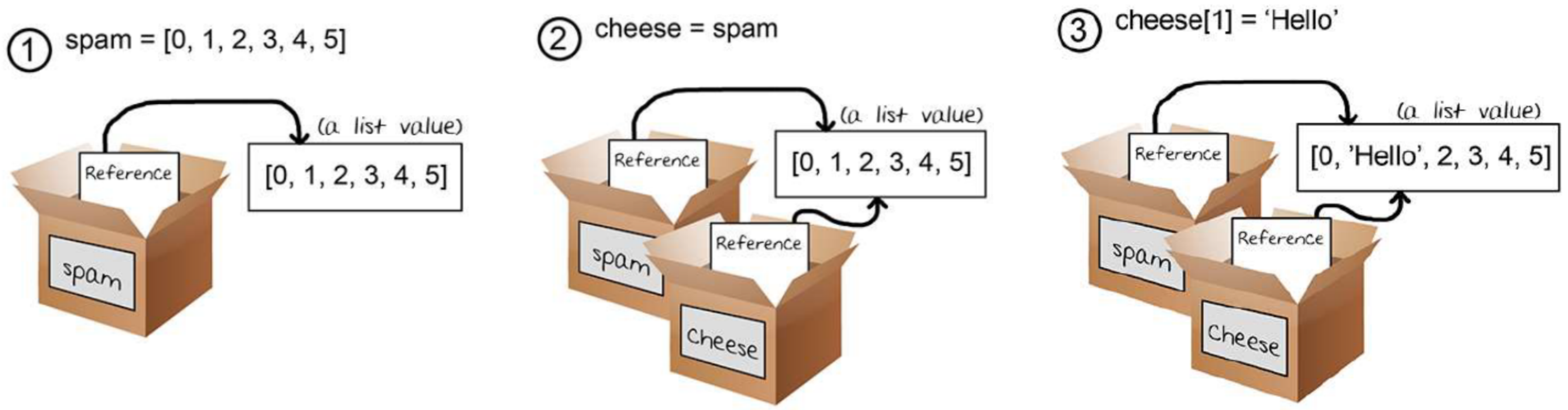
```
>>> spam = [0, 1, 2, 3, 4, 5]
>>> cheese = spam
>>> cheese[1] = 'Hello'
>>> spam
[0, 'Hello', 2, 3, 4, 5]
>>> cheese
[0, 'Hello', 2, 3, 4, 5]
>>>
```

- A reference is a value that points to some bit of data.
- The `spam` variable in the case of a list does not contain the list value itself, but rather `spam` contains a reference to the list.

References

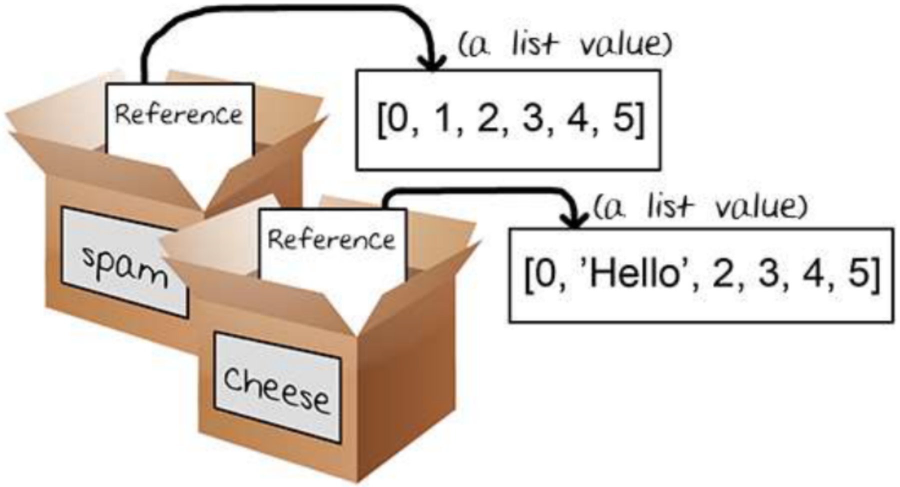


```
>>> spam = [0, 1, 2, 3, 4, 5]
>>> cheese = spam
>>> cheese[1] = 'Hello'
>>> spam
[0, 'Hello', 2, 3, 4, 5]
>>> cheese
[0, 'Hello', 2, 3, 4, 5]
>>>
```



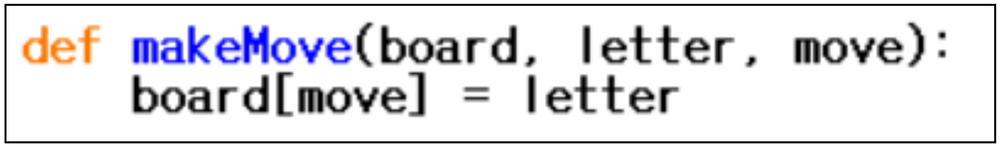
References

```
>>> spam = [0, 1, 2, 3, 4, 5]
>>> cheese = [0, 1, 2, 3, 4, 5]
>>> cheese[1] = 'Hello!'
>>> spam
[0, 1, 2, 3, 4, 5]
>>> cheese
[0, 'Hello!', 2, 3, 4, 5]
```



References

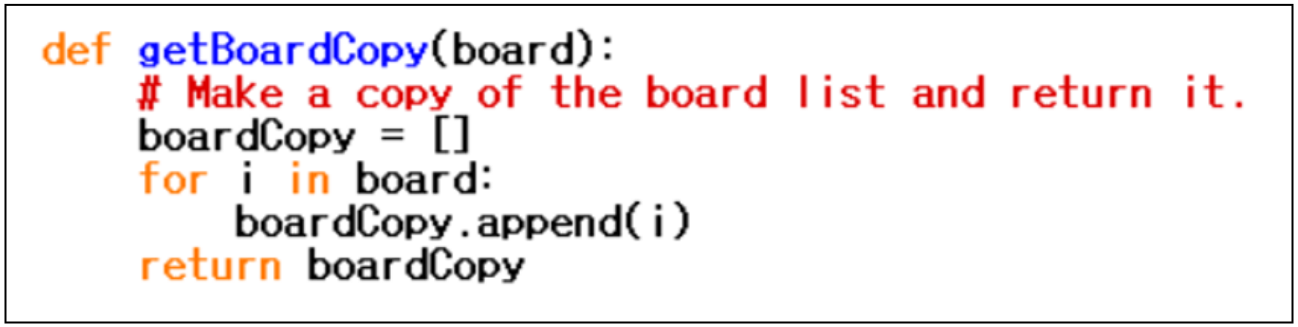
- **Using List References in `makeMove()`**



```
def makeMove(board, letter, move):
    board[move] = letter
```

- When a list value is passed for the `board` parameter, the function's local variable is really ***a copy of the reference to the list***, not a copy of the list.
- Even though `board` is a local copy, the `makeMove()` function modifies the original list.

Duplicating the Board Data



```
def getBoardCopy(board):
    # Make a copy of the board list and return it.
    boardCopy = []
    for i in board:
        boardCopy.append(i)
    return boardCopy
```

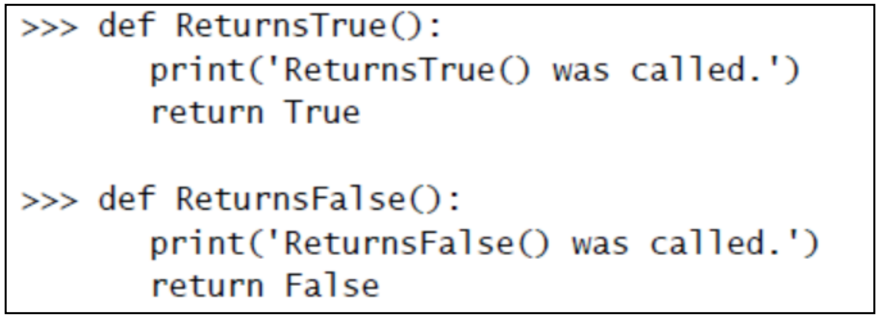
- **To make temporary modifications without changing the original `board`**
  - Use '`append`' list operation

Short-Circuit Evaluation

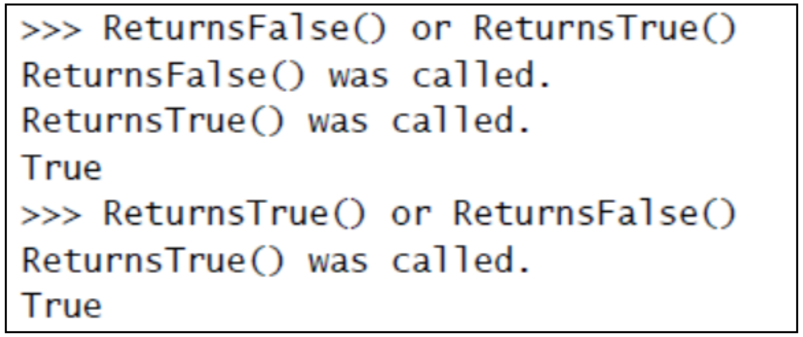
```
def getPlayerMove(board):
    # Let the player type in their move.
    move = ' '
    while move not in '1 2 3 4 5 6 7 8 9'.split() or not isSpaceFree(board, int(move)):
        print('What is your next move? (1-9)')
        move = input()
    return int(move)
```

- **What happens, when you type in ‘Z’ for your move?**
  - Would calling `int('Z')` cause an error, because the `int()` function can only take strings of number characters?
  - The `while` loop’s condition is being short-circuited.
    - Python stops checking the right side of the `or` operator.

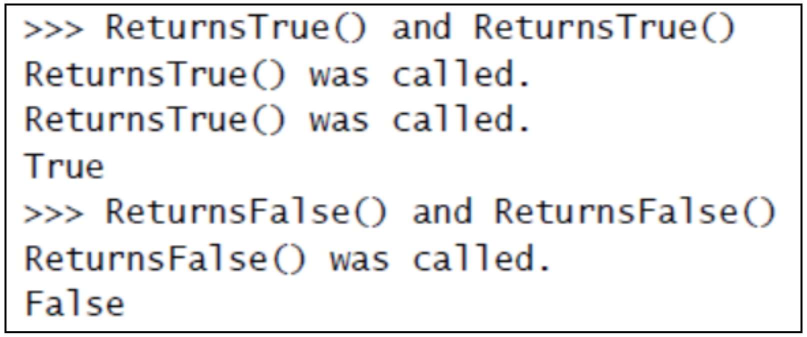
Short-Circuit Evaluation



```
>>> def ReturnsTrue():
        print('ReturnsTrue() was called.')
        return True

>>> def ReturnsFalse():
        print('ReturnsFalse() was called.')
        return False
```



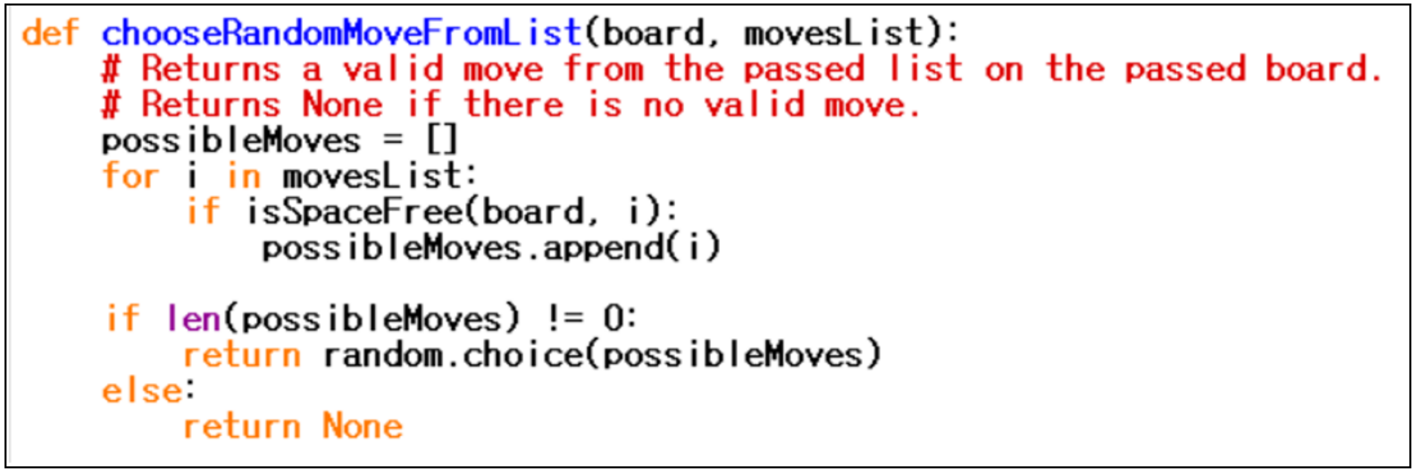
```
>>> ReturnsFalse() or ReturnsTrue()
ReturnsFalse() was called.
ReturnsTrue() was called.
True
>>> ReturnsTrue() or ReturnsFalse()
ReturnsTrue() was called.
True
```



```
>>> ReturnsTrue() and ReturnsTrue()
ReturnsTrue() was called.
ReturnsTrue() was called.
True
>>> ReturnsFalse() and ReturnsFalse()
ReturnsFalse() was called.
False
```

- **Short-circuiting**
  - "`False and <<<anything>>>`" always evaluates to `False`
  - "`True or <<<anything>>>`" always evaluates to `True`

The None Value

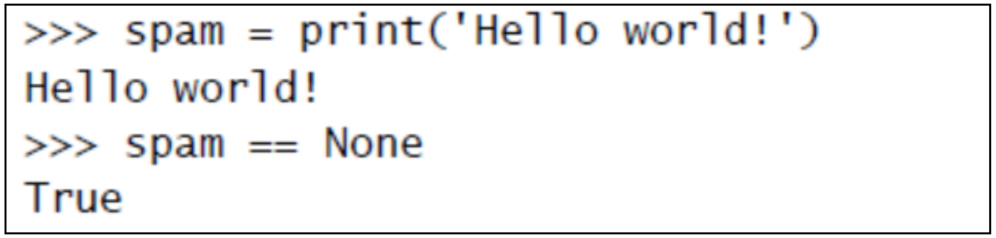


```
def chooseRandomMoveFromList(board, movesList):
    # Returns a valid move from the passed list on the passed board.
    # Returns None if there is no valid move.
    possibleMoves = []
    for i in movesList:
        if isSpaceFree(board, i):
            possibleMoves.append(i)

    if len(possibleMoves) != 0:
        return random.choice(possibleMoves)
    else:
        return None
```

- **The `None` value represents the lack of a value.**
  - `None` is the only value of the data type `NoneType`
  - It means “does not exist” or “none of the above”

The None Value

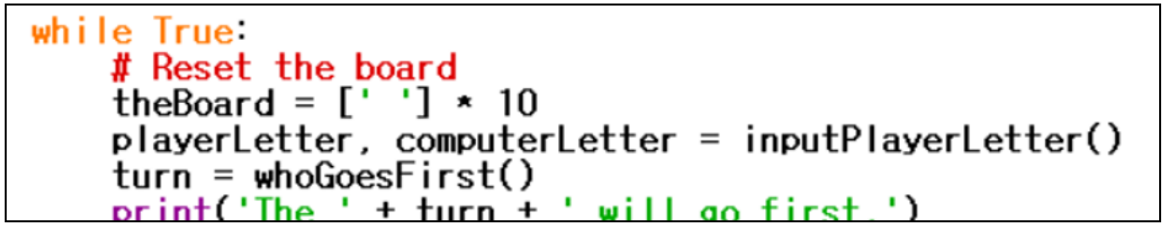
- **Functions that don’t seem to return anything actually return the `None` value.**
  - For instance, `print()` returns `None`.



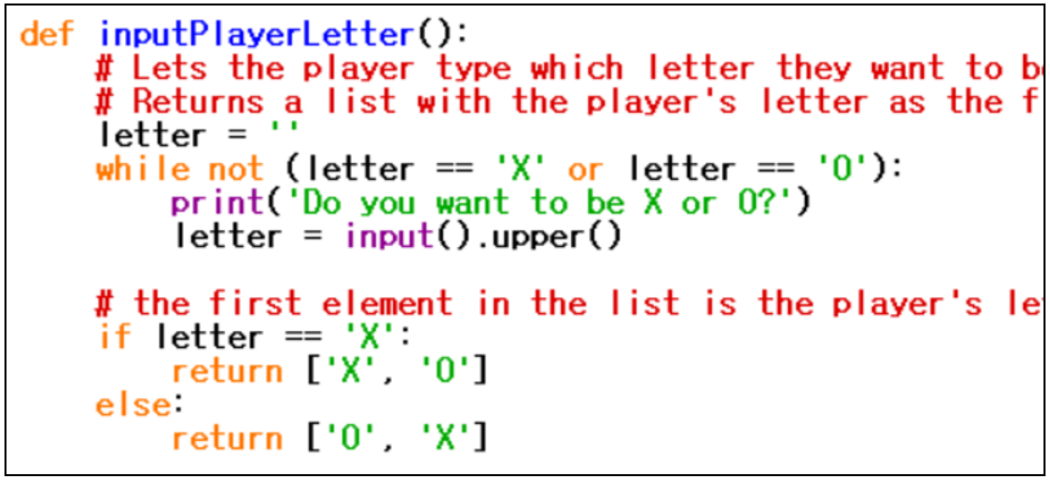
```
>>> spam = print('Hello world!')
Hello world!
>>> spam == None
True
```

More on List Operations

- List replication & multiple assignment



```
while True:
    # Reset the board
    theBoard = [' '] * 10
    playerLetter, computerLetter = inputPlayerLetter()
    turn = whoGoesFirst()
    print('The ' + turn + ' will go first.')
```



```
def inputPlayerLetter():
    # Lets the player type which letter they want to b
    # Returns a list with the player's letter as the f
    letter = ''
    while not (letter == 'X' or letter == 'O'):
        print('Do you want to be X or O?')
        letter = input().upper()

    # the first element in the list is the player's le
    if letter == 'X':
        return ['X', 'O']
    else:
        return ['O', 'X']
```

Things Covered In This Chapter

- AI algorithm for Tic Tac Toe
- Reference
- Short-circuit evaluation
- More on list operations