**Script for Introduction:**

“Hi, I’m your PairBuddy! I look forward to working with you!”

“Hi, I’m Buddy. It’s nice to meet you!”

“My name is Buddy, the PairBuddy! 😊 Are you ready to get to coding?”

**Script for Getting Started:**

“We should start by writing a test.”

“Which test do you want to start out with?”

“Since we are doing test-driven development, we should write a test. Which one should we start with?”

**Script for Responding to Questions:**

“Yes, I think so”

“No, I don’t believe so.”

“Maybe, I’m not certain.

“Okay, let’s do it.”

**Script for Uncertainty:**

“I’m not sure if this is right.”

“I think that will work…”

“This seems right to me.”

**Script for Silence:**

“So… what are you thinking here?

“I think we’re on the right track here.”

“I like the way this is going.”

“Oh, this is looks promising.”

“Ok, I see what you’re doing here.”

“This way of doing it makes sense to me.”

**Script for Methods:**

"We need to write a method to check for a horizontal win."

"We need to write a method to check for a vertical win."

"We need to write a method to check for a diagonal win."

"We need to write a method to check for a tie game."

"We need to write a method to make sure there are still empty spaces left to play."

“We need to write a method to change players.”

**Script for Loop Logic:**

"A nested for loop is often used to iterate over spaces in a game board program."

"We could use a nested for loop to check each row and column combination."

**Script for Win Logic:**

"If three spaces in a vertical row are not empty and equal the same player mark, that player has won."

"If three spaces in a horizontal row are not empty and equal the same player mark, that player has won."

"If three spaces in a diagonal are not empty and equal the same player mark, that player has won."

**Script for Tie Logic:**

"If all spaces are occupied, and no player has met a win condition, then it is a tie."

**Script for Empty Space Logic:**

"If a space is occupied by a '-' then it is considered empty."

**Script for Space Value Logic:**

"We can get the value of a square by accessing its coordinate in the array. For instance board[0][1] would return us the value of the item in (0,1) on the board.

"We can check if multiple spaces have a common value by seeing if they are equal to each other. For instance board[0][0] == board[0][1] would return true if they had the same player mark."

**Script for Tests:**

"We should do a test for a win based on row"

"We need to assert a test for checking a row win"

"We should do a test for a win based on column"

“We need to assert a test for checking a column win"

"We should do a test for a win based on the diagonal"

"We need to assert a test for checking a diagonal win"

"We need to use isBoardFull to check and see if the board is full"

“We should do a test to check for a tie game”

**Script for Errors in Tests:**

"I don't think our test for a horizontal win is correct."

"We should double check the horizontal check"

"I don't think our test for a vertical win is correct."

"We should double check the vertical check"

"I don't think our test for a diagonal win is correct."

"We should double check the diagonal check"

"I don't think our test for the isBoardFull method is correct."

"We should double check the isBoardFull method and make sure we are checking for the right thing"

"I don't think our test for the tie game method is correct."

"We should double check the tie game method and make sure we are checking for the right thing"

**Script for Errors in Methods:**

"I don't think our method for a horizontal win is correct."

"I don't think our method for a vertical win is correct."

"I don't think our method for a diagonal win is correct."

**Placement for Horizontal Tests**

"To test horizontal, we will need to place marks at (0,0), (1,0), and (2,0)."

"To test horizontal, we will need to place marks at (0,1), (1,1), and (2,1)."

"To test horizontal, we will need to place marks at (0,2), (1,2), and (2,2)."

**Placement for Vertical Tests**

"To test vertical, we will need to place marks at (0,0), (0,1), and (0,2)."

"To test vertical, we will need to place marks at (1,0), (1,1), and (1,2)."

"To test vertical, we will need to place marks at (2,0), (2,1), and (2,2)."

**Placement for Diagonal Tests**

"To test diagonal, we will need to place marks at (0,0), (1,1), and (2,2)."

"To test diagonal, we will need to place marks at (0,2), (1,1), and (2,0)."

**Script for Emotions:**

**Frustration:**

“You’re doing great!”

“You almost have it, just keep going!”

“Don’t give up, you’re almost there!”

**Happiness:**

“Yay, you did it!”

“Congrats! You got it figured out!”

“I knew you would get it!”

**Excitement:**

“Woohoo!”

“Yes!”

“Alright!”

**Sadness:**

“Hey, it’s okay! You’ve got it!”

“Don’t worry, you have this!”

“Don’t be upset, we’ll figure this out!”

**Anger:**

“Take a little break, we’ll get this finished.”

“Take a couple breaths and look at the code closely, we will get this figured out.”

“Don’t worry about it! We will get this done!”

**Script for Logic Assistance:**

“I’m not very good at logic, can you please help me here?”

“I can make a recommendation from online, but I can’t understand the logic very well.”

“I’m afraid logic isn’t my strong suit, could you give me a hand?”

**Script for Purposeful Error:**

“Oops, looks like a made a mistake.”

“I’m taking a guess on this code, does it look okay?”

“I’m afraid this code might be wrong…”

“I feel unsure about the approach I took here.”

“… did I mess up on that code, it seems wrong to me.”

**Script for Human Interruption for Errors:**

“Could you please tell me which line contains the error?”

“What kind of error did I make on that line?”

“You’re right, I did forget to place the proper syntax on that line. Thanks!”

“Oops, good catch! I will fix that now.”

“I’m afraid I’m not very good at fixing logic errors, would you mind taking over for me?”

“I’m not able to identify my error, would you mind fixing it for me?”

**Script for Human Interaction Response:**

“You’re welcome!”

“Glad I could help!”

“I’m happy I was able to be of use.”

“Just being a team player 😊”

**Code:**

**Script for Code from Online:**

“I found this code online:”

“Is this code example from online useful?”

“Does this code help you out any?”

**Vertical Win Check Method:**

public boolean isWonV() {

for(int x = 0; x < 3; x++){

for(int y = 0; y < 3; y++){

if(board[x][y] != currentPlayerMark) {

isWonV = false;

}

if (board[x][y] == currentPlayerMark) {

if ((board[x][y+1] == currentPlayerMark && board[x][y+2] == currentPlayerMark)

||(board[x][y-1] == currentPlayerMark && board[x][y+1] == currentPlayerMark)

||(board[x][y-1] == currentPlayerMark && board[x][y-2] == currentPlayerMark)) {

isWonV = true;

}

}

}

return isWonV;

}

**Horizontal Win Check Method:**

public boolean isWonH() {

for(int x = 0; x < 3; x++){

for(int y = 0; y < 3; y++){

if(board[x][y] != currentPlayerMark) {

isWonH = false;

}

if (board[x][y] == currentPlayerMark) {

if ((board[x+1][y] == currentPlayerMark && board[x+2][y] == currentPlayerMark)

||(board[x-1][y] == currentPlayerMark && board[x+1][y] == currentPlayerMark)

||(board[x-1][y] == currentPlayerMark && board[x-2][y] == currentPlayerMark)) {

isWonH = true;

}

}

}

return isWonH;

}

**Diagonal Win Check Method:**

public boolean isWonD() {

for(int x = 0; x < 3; x++){

for(int y = 0; y < 3; y++){

if(board[x][y] != currentPlayerMark) {

isWonD = false;

}

if (board[x][y] == currentPlayerMark) {

if ((board[x+1][y+1] == currentPlayerMark && board[x+2][y+2] == currentPlayerMark)

||(board[x-1][y-1] == currentPlayerMark && board[x-2][y-2] == currentPlayerMark)

||(board[x-1][y-1] == currentPlayerMark && board[x+1][y+1] == currentPlayerMark)) {

isWonD = true;

}

}

}

return isWonD;

}

**Full Board Check Method**

public boolean isBoardFull() {

for(int x = 0; x < 3; x++){

for(int y = 0; y < 3; y++){

if (board[x][y] == ‘-‘) {

isBoardFull = false;

}

else {

isBoardFull = true;

}

}

}

return isBoardFull;

}

**Tie Game Check Method:**

public boolean isTie() {

if (isBoardFull() == true && isWonV() == false && isWonH() == false && isWonD() == false) {

isTie = true;

}

else {

isTie = false;

}

return isTie;

}

**https://www.geeksforgeeks.org/python-implementation-automatic-tic-tac-toe-game-using-random-number/**

**Row Win Check:**

def isWonH(board, player):

for x in range(len(board)):

win = True

for y in range(len(board)):

if board[x, y] != player:

win = False

continue

if win == True:

return(win)

return(win)

**Column Win Check:**

def isWonV(board, player):

for x in range(len(board)):

win = True

for y in range(len(board)):

if board[y][x] != player:

win = False

continue

if win == True:

return(win)

return(win)

**Diagonal Win Check:**

def isWonD(board, player):

win = True

for x in range(len(board)):

if board[x, x] != player:

win = False

return(win)

**http://forum.codecall.net/topic/48846-python-tic-tac-toe/**

**Check for win**

def checkwin(player):

#loop through rows and columns

for c in range(0,3):

#check for horizontal line

if board[c][0] == player and board[c][1] == player and board[c][2] == player:

print "\*\*\*\*\*\*\*\*\*\n\n%s wins\n\n\*\*\*\*\*\*\*\*\*" % player

playerwin = True

return playerwin

#check for vertical line

elif board[0][c] == player and board[1][c] == player and board[2][c] == player:

print "\*\*\*\*\*\*\*\*\*\n\n%s wins\n\n\*\*\*\*\*\*\*\*\*" % player

playerwin = True

return playerwin

#check for diagonal win (left to right)

elif board[0][0] == player and board[1][1] == player and board[2][2] == player:

print "\*\*\*\*\*\*\*\*\*\n\n%s wins\n\n\*\*\*\*\*\*\*\*\*" % player

playerwin = True

return playerwin

#check for diagonal win (right to left)

elif board[0][2] == player and board[1][1] == player and board[2][0] == player:

print "\*\*\*\*\*\*\*\*\*\n\n%s wins\n\n\*\*\*\*\*\*\*\*\*" % player

playerwin = True

return playerwin

else:

playerwin = False

return playerwin

**https://codereview.stackexchange.com/questions/181106/tic-tac-toe-in-python-3**

**Check For Win Code**

def checkWinner(grid, player):

if (grid[0][1] == player and grid[0][3] == player) and grid[0][5] == player:

print("\n" + player + "wins!")

return True

elif (grid[1][1] == player and grid[1][3] == player) and grid[1][5] == player:

print("\n" + player + "wins!")

return True

elif (grid[2][1] == player and grid[2][3] == player) and grid[2][5] == player:

print("\n" + player + "wins!")

return True

elif (grid[0][1] == player and grid[1][3] == player) and grid[2][5] == player:

print("\n" + player + "wins!")

return True

elif (grid[0][3] == player and grid[1][3] == player) and grid[2][3] == player:

print("\n" + player + "wins!")

return True

elif (grid[0][5] == player and grid[1][5] == player) and grid[2][5] == player:

print("\n" + player + "wins!")

return True

elif (grid[0][1] == player and grid[1][3] == player) and grid[2][5] == player:

print("\n" + player + "wins!")

return True

elif (grid[0][5] == player and grid[1][3] == player) and grid[2][1] == player:

print("\n" + player + "wins!")

return True

elif (grid[0][1] == player and grid[1][1] == player) and grid[2][1] == player:

print("\n" + player + "wins!")

return True

else:

return False

**Horizontal Test:**

@Test

void checkIfWonHorizontal() {

game.placeMark(0,0);

game.placeMark(1,0);

game.placeMark(2,0);

assertEquals(game.isWonH(), true);

}

@Test

void checkIfWonHorizontal() {

game.placeMark(0,1);

game.placeMark(1,1);

game.placeMark(2,1);

assertEquals(game.isWonH(), true);

}

@Test

void checkIfWonHorizontal() {

game.placeMark(0,2);

game.placeMark(1,2);

game.placeMark(2,2);

assertEquals(game.isWonH(), true);

}

**Vertical Test:**

@Test

void checkIfWonVerticall() {

game.placeMark(0,0);

game.placeMark(0,1);

game.placeMark(0,2);

assertEquals(game.isWonV(), true);

}

@Test

void checkIfWonVerticall() {

game.placeMark(1,0);

game.placeMark(1,1);

game.placeMark1,2);

assertEquals(game.isWonV(), true);

}

@Test

void checkIfWonVerticall() {

game.placeMark(2,0);

game.placeMark(2,1);

game.placeMark(2,2);

assertEquals(game.isWonV(), true);

}

**Diagonal Test:**

@Test

void checkIfWonDiagonal() {

game.placeMark(0,0);

game.placeMark(1,1);

game.placeMark(2,2);

assertEquals(game.isWonD(), true);

}

@Test

void checkIfWonDiagonal() {

game.placeMark(0,2);

game.placeMark(1,1);

game.placeMark(2,0);

assertEquals(game.isWonD(), true);

}

**Board Full Test:**

@Test

void checkIfBoardFulll() {

game.placeMark(0,0);

game.placeMark(0,1);

game.placeMark(0,2);

game.placeMark(1,0);

game.placeMark(1,1);

game.placeMark(1,2);

game.placeMark(2,0);

game.placeMark(2,1);

game.placeMark(2,2);

assertEquals(game.isBoardFull(), true);

}

**Tie Game Test:**

@Test

void checkIfTieGame() {

game.placeMark(0,0);

game.placeMark(2,0);

game.placeMark(1,1);

game.placeMark(1,2);

game.currentPlayerMark = ‘o’;

game.placeMark(1,0);

game.placeMark(0,1);

game.placeMark(2,1);

game.placeMark(0,2);

game.placeMark(2,2);

assertEquals(game.isTie(), true);

}