CMP5361 COMPUTATIONAL MATHEMATICS AND DECLARATIVE PROGRAMMING

LOG REPORT

Dhananjay Tewari

21150306

Rules of tic tac toe game:

Tic Tac toe also known as noughts(Os) and crosses(Xs) is a two player game as by the name containing the token ‘X’ and ‘O’.

Played in turns marking the spaces in three by three grid with their token, during their turn.

The player who succeeds to place three of their token together horizontally, vertically or diagonally is the winner. When both of the player unable to do so the game is forced to draw and is called the solved game.

Requirements:

Allow the Player to start a new 2-Player or single Player game with each player taking turns will be judged by the program itself. Also allow the player to select their desire token at the start of the game and select where to play the token on the board.

User Specification:

Gherkin Specification:

Feature: Menu

Scenario: Selecting the Player VS Player Option

Given the game is started

And Main Menu Appear

When option with Player vs Player Selected by stdin

Then Game asks to select the Token

And Game Started

Scenario: Selecting the Player Vs Computer Option

Given The Game Started

And Main Menu Appeared

When option with player VS computer selected by stdin

Then Game asks to select the Token

And Game began

Scenario: Selecting the Quit Option

Given The Game Started

And Main Menu appeared

When option to quit Selected by stdin

Then Good Bye message Appeared

And Game Began

Scenario: Selecting the Invalid Key

Given The Game Started

And Main Menu appeared

When option to invalid option selected by stdin

Then "Invalid Input" Message appeared

And Asked For the Input Again

Feature: Playing a Token

Scenario: Selecting the Valid Coordinates and Selected Cell is Empty

Given The Game Started

And Main Menu Appeared

And Player Vs Player option Selected through stdin

And Token Selected

When put Valid Coordinated of Empty cell to place the token

Then Token is Placed on the Board

And Displayed on the Board

Scenario: Selecting the Valid Coordinates and Selected Cell is Filled

Given The Game Started

And Main Menu Appeared

And Player Vs Player option Selected through stdin

And Token Selected

When put Valid Coordinated of filled cell to place the token

Then "Token already placed" message appeared on stout

And Ask to give the Coordinates again

Scenario: Selecting the Invalid Coordinates

Given The Game Started

And Main Menu Appeared

And Player Vs Player option Selected

And Token Selected

When put invalid Coordinated

Then "Invalid Coordinate " message appeared on Stdout

And game asked for new stdin Coordinates

Feature: Selecting Token

Scenario: Selecting the X Token

Given The Game Started

And Main Menu Appeared

And Player Vs Player option Selected

And Stdout Display ask for the Token

When Token "X" Selected

Then Player 1 got Token X

And Player 2 got Token O

And game asked on stdin Coordinates of Cell

Scenario: Selecting the O Token

Given The Game Started

And Main Menu Appeared

And Player Vs Player option Selected

And Stdout Display ask for the Token

Wh**en** Token "O" Selected

Then Player 1 got Token O

And Player 2 got Token X

And game asked on stdin Coordinates of Cell

Scenario: Selecting the Invalid Token

Given The Game Started

And Main Menu Appeared

And Player Vs Player option Selected

And Stdout Display ask for the Token

When invalid token Selected

Then message appears on stdout “Invalid token”

And System asks for another stdin Token

Feature: Checking The Play

Scenario: Checking the Play when the game is tied

Given The Game Started

And Main Menu Appeared

And Player Vs Player option Selected

And Stdout Display ask for the Token

And Token “O” Selected

And Player 1 got Token O

And Player 2 got Token X

And game asked on stdin Coordinates of Cell

When Board gets filled with Tokens with no Winner

Then board is checked

And Game result=Tied

Scenario: Checking the play when game is Won by a Token

Given The Game Started

And Main Menu Appeared

And Player Vs Player option Selected

And Stdout Display ask for the Token

And Token “O” Selected

And Player 1 got Token O

And Player 2 got Token X

And game asked on stdin Coordinates of Cell

When player 1 succeeded to place three of his token diagonally

Then board is checked

And Game result=O win

Modelling:

Data Modelling:

Token

The Token Used in the game of the tic tac toe is should be created as the data in program to access and get used for the play so for that the new data type was created called Token which has X and O as its elements will be considered as the type of Token.

Token: Datatype: X or O

Token={X}U{O}

Token={X,O}

Cell

For creating the block in which player places the token Cell Data type is created which consists of the Token as its element or nothing in it which can be said null so for that Data Type Cell is named version of <optional> data type of token.

Cell=X or O or None

Cell=Token U {None}

Cell=optional<Token>

Row

By creating the pair of three Cell type 1 Row of the Tic tac Toe board is created which is the product three Cell set. This type of data structure should be stored in the data structure which can allow the user to store only three element of type Cell. So for that this Row data structure is being created as the typed Tuple of type cell.

Row =CellXCellXCell

Row=(Cell,Cell,Cell)

Grid

To create the whole board for the game of tic tac toe which will be the Product of three Row type. The Grid is the typed Tuple of Type Row which Consists of three Row Data type as the Row can be said as the 2d Tuple.

Grid=RowXRowXRow

Grid=(Row,Row,Row)

Coordinates

To access the each and every element of the Grid data structure some coordinates are required to access is which is Stored in Coordinate data type which is further divide into two sub types Horizontal Coordinates to get the location horizontally of each elements getting a full row and Vertical Coordinates to get the elements Vertically from the grid getting the full Column.

Axis:

As the Grid Consists of three column the Vertical Coordinates will consists of three elements defining each Column.

Axis={A,B,C}

Combining the vertical coordinates and horizontal coordinates you will get the Coordinate of the specific cell.

Cord=Axis X Axis

Cord={(A,A),(A,B),(A,C),(B,A),(B,B),(B,C),(C,A),(C,B),(C,C)}

Accessing the elements (typed dictionary):

Accessing the element From the grid by each coordinate is linked together using the Typed Dictionary where every Co-ordinates is a key and Cell is value making the key cell value pair for the Accessing the elements of the grid.

Access=TypedDictionary<Coordinates, Cell>

Access={(x ϵ Coordinates and y ϵ Cell|(x,y)}

Method Modelling:

Parsing the String to the valid Token:

Parsing the String is Done to Convert or Parse the String Input to the Function into a valid Token

$$\frac{parseToken: String \rightarrow Token}{\begin{gathered} parseToken\ \text{"X"} = X \\ parseToken\ \text{"O"} = O \\ parseToken\ \text{"x"} = X \\ parseToken\ \text{"o"} = O \end{gathered}}$$

If I were to Define the all Input and Output pairs for the above function it would look like this:

$$\frac{parseToken : String\ \times Token}{allBoolPairs = \ \{(\text{"}x\text{"}, X), (\text{"}X\text{"}, X), (\text{"}O\text{"}, O), (\text{"}o\text{"}, O)\}}$$

This Function is the pure Function as it return the same argument value and it has no side effects.

To make this function Total Function if any other string is been input other than “X”,”x”,”O,”o” than it returns None. So For every element of the Domain there is return value in the Range hence making it a Total Function.

Parsing the String to the valid Coordinate Axis

Parsing the String is Done To Convert given String in the Source Set into the Target set which is Strings and Coordinates Respectively

$$\frac{parseCoordinatesAxis: String \rightarrow Axis}{\begin{gathered} parseCoordinatesAxis\ \text{"a"} = A \\ parseCoordinatesAxis\ \ \text{"A"} = A \\ parseCoordinatesAxis\ \text{"B"} = \ B \\ parseCoordinatesAxis\ \text{"b"} = B \\ parseCoordinatesAxis\ \text{"c"} = C \\ parseCoordinatesAxis\ \text{"C"} = C \end{gathered}}$$

If I were to Define the all Input and Output pairs for the above function it would look like this:

$$\frac{parseCoordinatesAxis : String \rightarrow\ \text{Axis}}{parseCoordinatesAxis = \{(\text{"a"}, A), (\text{"A"},\text{A}), (\text{"b"}, B), (\text{"B"}, B), (\text{"c"}, C), (\text{"C"}, C)\}}$$

This Function is the pure Function as it return the same argument value and it has no side effects.

To make this function Total Function if any other string is been input other than “A”,”a”,”B,”b”,”c”,”C” than it returns None. So For every element of the Domain there is return value in the Range hence making it a Total Function.

Getting the Token

Get Token is the function which has Void in the Domain but Return the Token as the Result in the Range after Parsing it using the parse Token Function.

$$\frac{getToken :\ \text{Void} \rightarrow \text{Token}}{getToken = \{x :\ \text{Void}\ \cdot (\text{Void}, \text{parseToken stdin})\}}$$

Where stdin is the standard input From the user

As the output of the function depends on the external pattern it is impure function.

Getting the Coordinate Axis

Get Coordinates Axis To get the String From the User and Returning the Axis by Parsing the given input.

$$getCoordinatesAxis: Void \rightarrow Axis$$

As the output of the function depends on the external pattern it is impure function.

Getting the Coordinate

To get the Coordinate Tuple of Axis the get Coordinates Axis Function is used to get the Axis for the Vertical and Horizontal side as well. So by getting that the Tuple of the (Axis, Axis) which is the Type coordinate.

$$getCoordinates: Void \rightarrow Coordinate$$

As the output of the function depends on the external pattern it is impure function.

Create Board From Access

Create Board Function Creates the Board From the given Access. So It has Access in the Source Set and Board in the Target Set

$$createBoard: Access \rightarrow Board$$

$$\frac{createBoard: Access \rightarrow Board}{createBoard = \{x : Access\ and\ y: Board \cdot (\mathrm{x}, \mathrm{y})\}}$$

It is pure function as the input and output did not depends on any external factor and gives same output for the same input.

It is also Total Function because for Every Access type in domain there is Board type in range

Create Access From Board

Output the access From the given input which is board .Creating the Domain of element of Type Board and Range of Type Access.

$$\frac{createAccess : \ \mathrm{Board} \rightarrow \ \mathrm{Access}}{createAccess = \ x: Board \land y: Access \cdot \lambda \mathrm{x} \ \cdot \mathrm{y}}$$

It is pure function as the input and output did not depends on any external factor.

It is also Total Function because for Every Board type in domain there is Access type in range

Play Token

Play token Takes Token and Board as the input and uses get Coordinates to get coordinates of the Token on the Board and then return the updated board with the token being placed at the given Coordinates

$$playToken: (Token, Board) \rightarrow Board$$

This is impure Function because it depends on the external factor like get coordinates.

Empty Board

Empty Board Takes no Input and return the Empty Board as the Output.

$$emptyBoard: Void \rightarrow Board$$

Check Tie Game

The check tie function gets the board and check if its fully if it is than the game is finished and tells us True or false rather the game is tied.

$$\frac{CheckTieGame: \text{ Board} \rightarrow \text{ Boolean}}{CheckTieGame = \{(board, True), (board, False)\}}$$

Where board is Type Board

It is Pure Function As it does not depends on the external factor always return the same output for the same given input

Display Board

Display Board take Board as the input but rather than returning the board it prints out the board on the screen in the given format.

$$Displayboard: Board \rightarrow Void$$

It is the partial function as there is not defined input for the elements of the domain

Display Menu

Display Menu is Void Function takes nothing in the input and returns nothing in the output just to print out the menu on the Screen

$$displayMenu: Void \rightarrow Void$$

It is the partial function as there is not defined input for the elements of the domain

Check Diagonals

Check Diagonals function take Access of the Board or Type Access as the input and return the Cell (Optional<Token>)type as output.

$$checkDiagnals: Access \rightarrow Cell$$

It is Total and Pure function as it has For all Access in domain there is cell in range and give same range element for same Domain element.

Check Row

Check Row function take Access as the input for accessing the Board and return Cell(Optional<Token>) as the Output.

$$checkRow: Access \rightarrow Cell$$

It is Total and Pure function as it has For all Access in domain there is cell in range and give same range element for same Domain element.

Check Column

Check Row function take Access as the input for accessing the Board and return Cell(Optional<Token>) as the Output.

$$checkRow: Access \rightarrow Cell$$

It is Total and Pure function as it has For all Access in domain there is cell in range and give same range element for same Domain element.

Check Winner

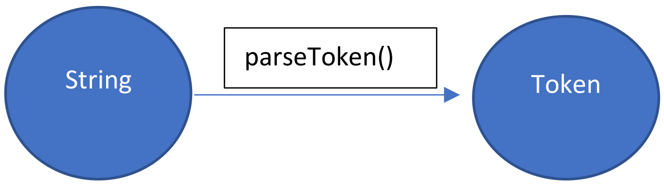
By checking the Board using the Access and getting the output using check Row, check diagonal and check column as the Cell.

$$checkWinner: Access \rightarrow Cell$$

It is Total and Pure function as it has For all Access in domain there is cell in range and give same range element for same Domain element.

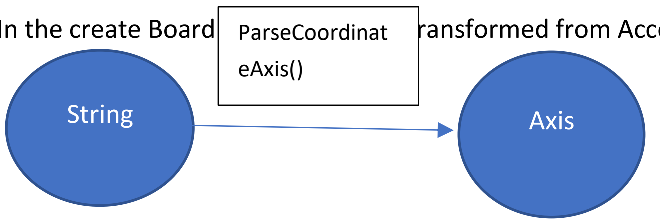
Data Transformation graph:

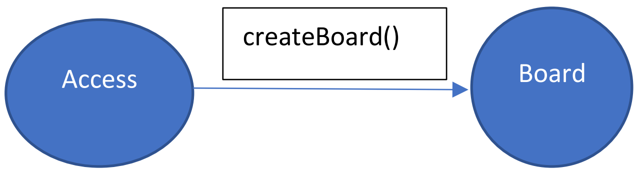
In the parse Token function the Data of the String data type is being Converted to the token Data Type.



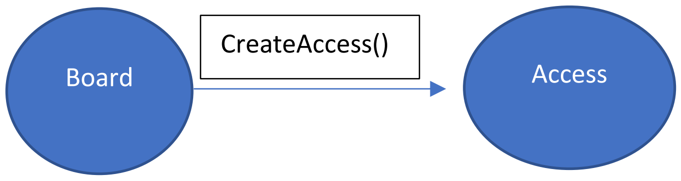
In the parse Coordinate Axis method the data of the Type string is converted into the data of type Axis.

In the create Board ransformed from Access to the Type Board



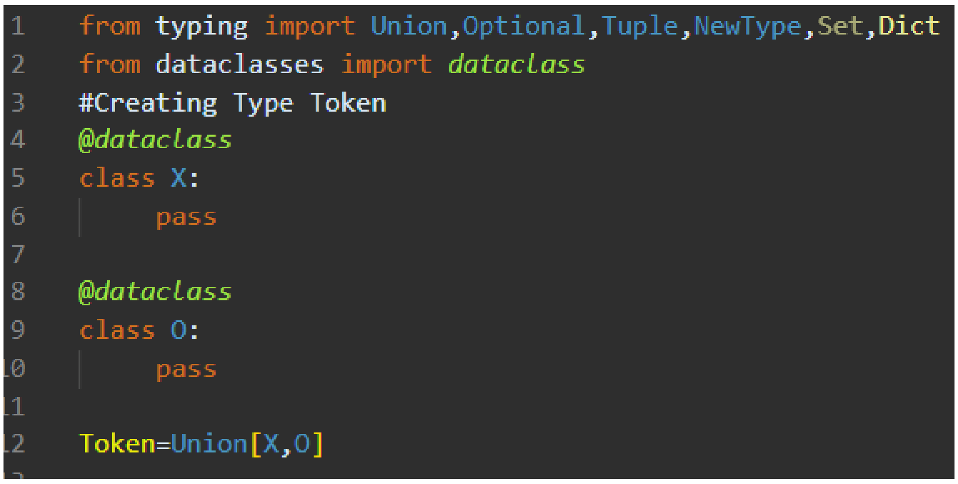


In the create Access Function Data is Transformed from Board to the type Access

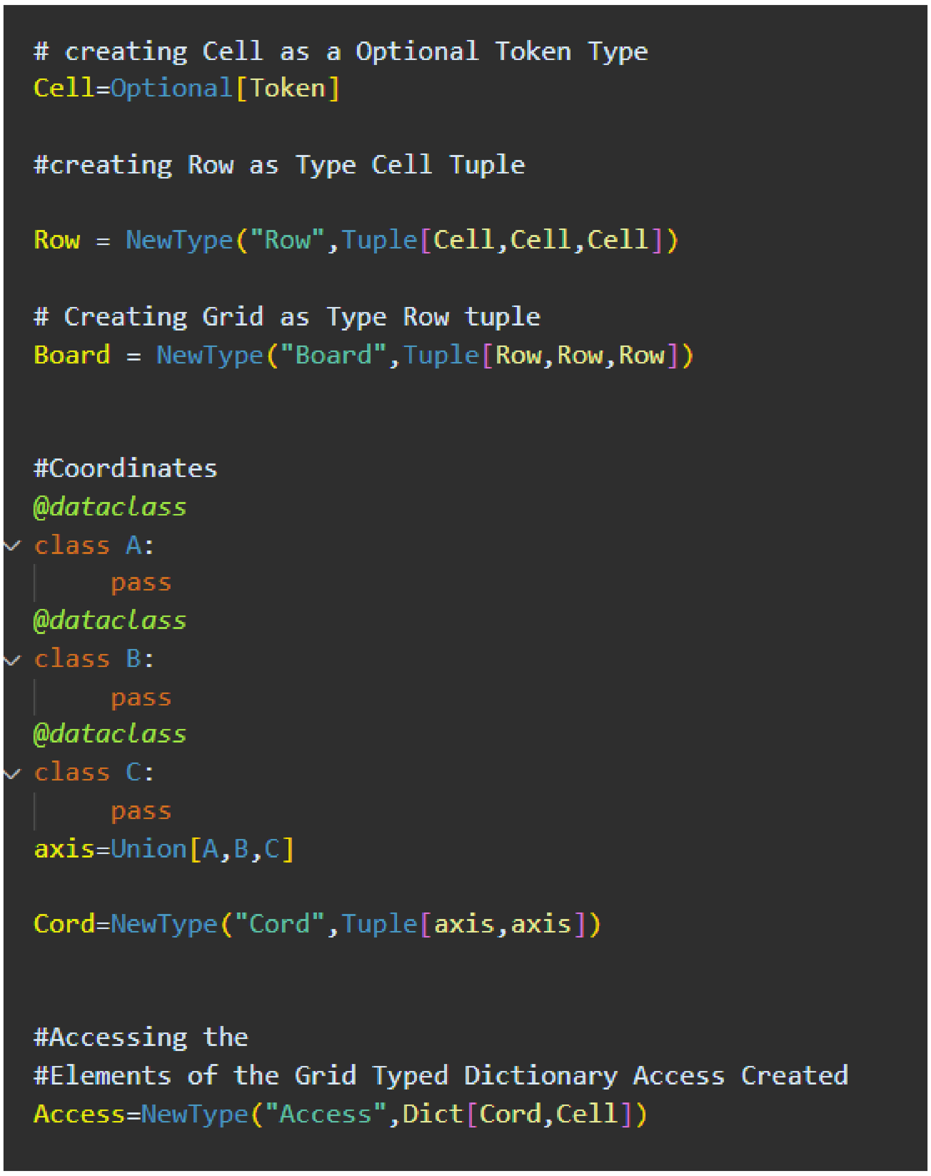


Implementation Discussion:

Creating the Data Types:



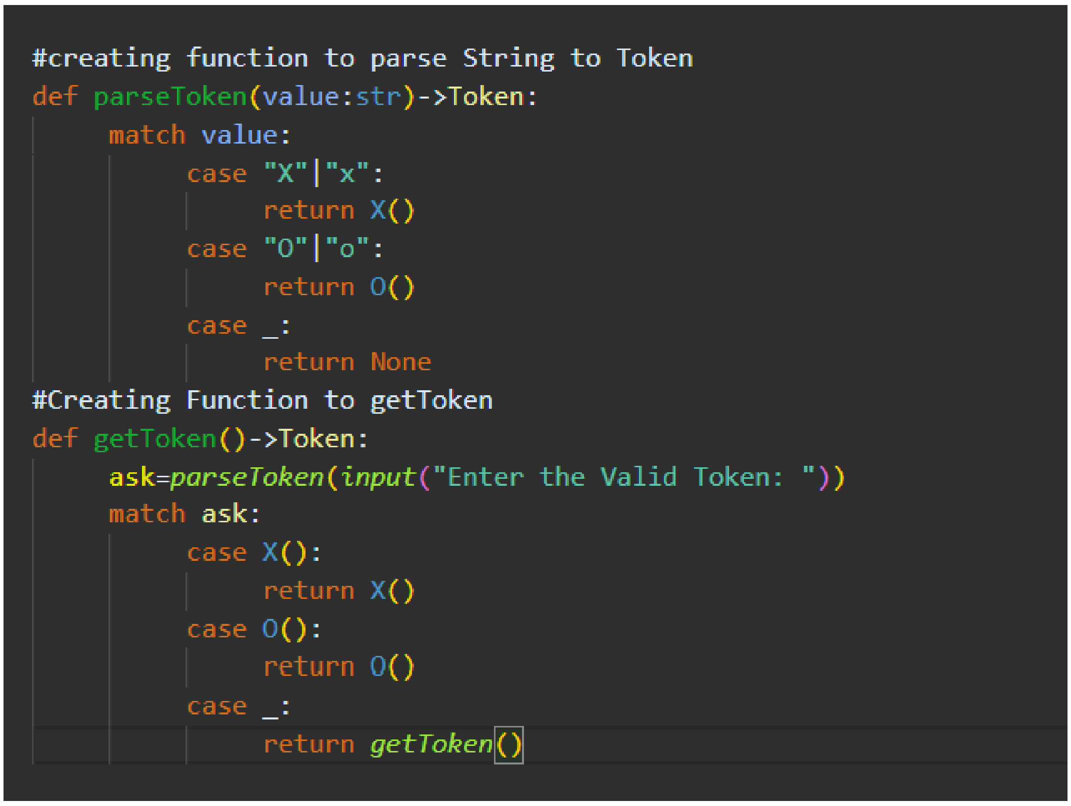
```python
from typing import Union,Optional,Tuple,NewType,Set,Dict
from dataclasses import dataclass
#Creating Type Token
@dataclass
class X:
        pass

@dataclass
class O:
        pass

Token=Union[X,O]
```



```python
# creating Cell as a Optional Token Type
Cell=Optional[Token]

#creating Row as Type Cell Tuple

Row = NewType("Row",Tuple[Cell,Cell,Cell])

# Creating Grid as Type Row tuple
Board = NewType("Board",Tuple[Row,Row,Row])


#Coordinates
@dataclass
class A:
        pass
@dataclass
class B:
        pass
@dataclass
class C:
        pass
axis=Union[A,B,C]

Cord=NewType("Cord",Tuple[axis,axis])


#Accessing the
#Elements of the Grid Typed Dictionary Access Created
Access=NewType("Access",Dict[Cord,Cell])
```

The new Types were created for the functions to limit their domain which will helps in Totalising the function making it easier to handle error.

Parsing and Getting Token From the String:



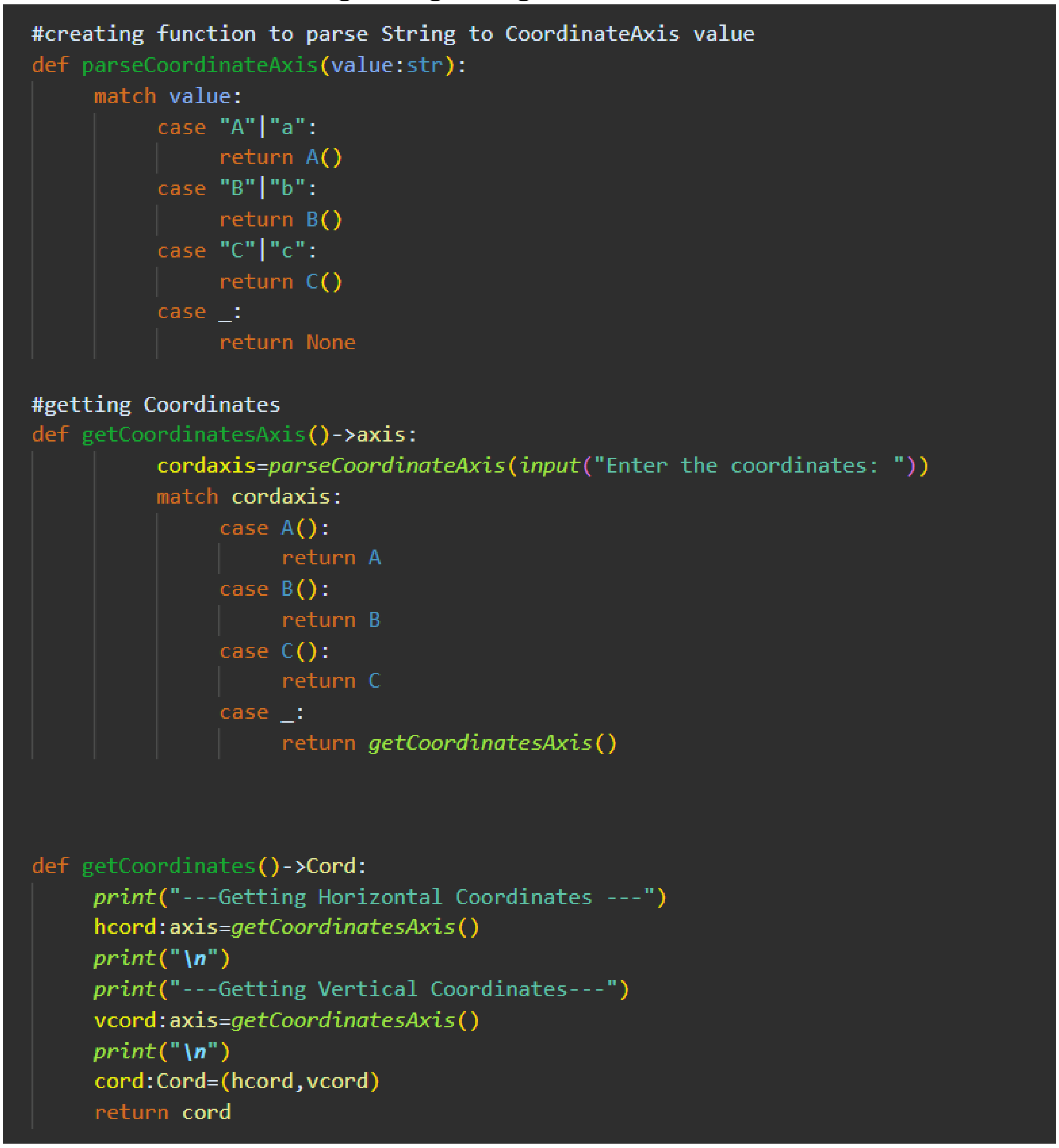
```python
#creating function to parse String to Token
def parseToken(value:str)->Token:
    match value:
        case "X"|"x":
            return X()
        case "O"|"o":
            return O()
        case _:
            return None
#Creating Function to getToken
def getToken()->Token:
    ask=parseToken(input("Enter the Valid Token: "))
    match ask:
        case X():
            return X()
        case O():
            return O()
        case _:
            return getToken()
```

Parsing the Token helps to get the input From the user in the form the String for their desired token and can be parsed into the Token of the Game and can be used for playing.

So the parseToken method takes String type as the input and return token as the output so which is used to parse the string input collected from the user in the getuser method than returning the valid Token after the interaction with user.

Here the range got Limited to the set of elements in Token and range become P(Token) power set of the Token. Also can be said as the <Optional>Token

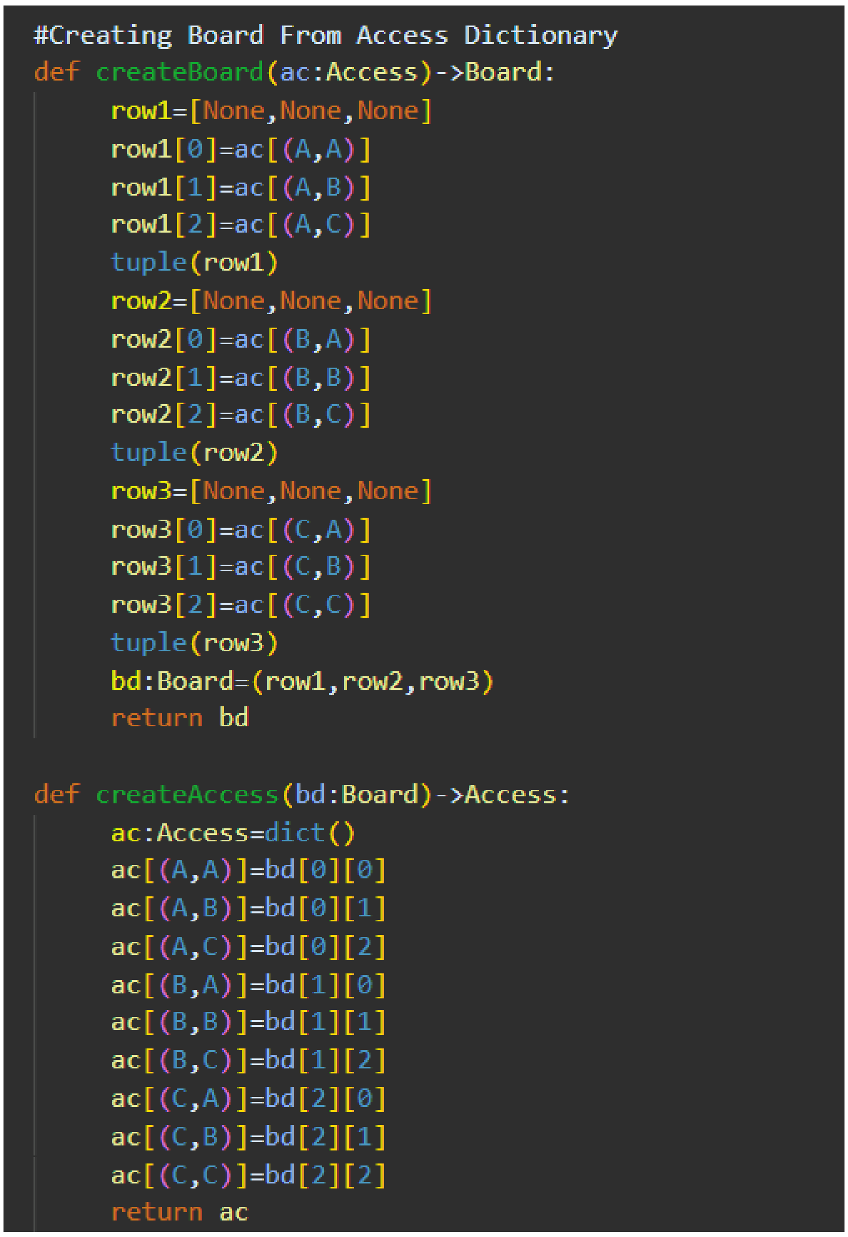
Parsing and getting Axis/Coordinates :



```python
#creating function to parse String to CoordinateAxis value
def parseCoordinateAxis(value:str):
    match value:
        case "A"|"a":
            return A()
        case "B"|"b":
            return B()
        case "C"|"c":
            return C()
        case _:
            return None

#getting Coordinates
def getCoordinatesAxis()->axis:
        cordaxis=parseCoordinateAxis(input("Enter the coordinates: "))
        match cordaxis:
            case A():
                return A
            case B():
                return B
            case C():
                return C
            case _:
                return getCoordinatesAxis()



def getCoordinates()->Cord:
    print("---Getting Horizontal Coordinates ---")
    hcord:axis=getCoordinatesAxis()
    print("\n")
    print("---Getting Vertical Coordinates---")
    vcord:axis=getCoordinatesAxis()
    print("\n")
    cord:Cord=(hcord,vcord)
    return cord
```

Parsing the axis helps to get the axis from the string which we got from the user interaction and using that horizontal axis and vertical axis to make a valid coordinate of the cell.

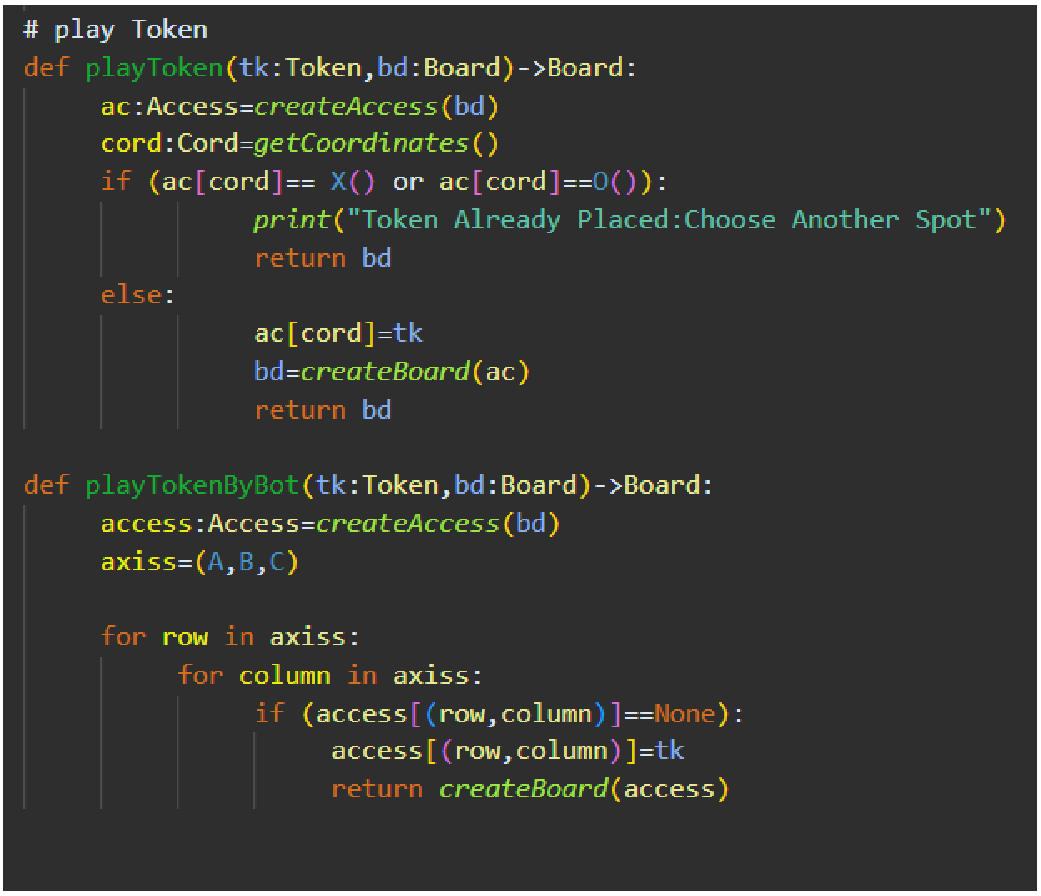
Creating Board and Access:



```
#Creating Board From Access Dictionary
def createBoard(ac:Access)->Board:
        row1=[None,None,None]
        row1[0]=ac[(A,A)]
        row1[1]=ac[(A,B)]
        row1[2]=ac[(A,C)]
        tuple(row1)
        row2=[None,None,None]
        row2[0]=ac[(B,A)]
        row2[1]=ac[(B,B)]
        row2[2]=ac[(B,C)]
        tuple(row2)
        row3=[None,None,None]
        row3[0]=ac[(C,A)]
        row3[1]=ac[(C,B)]
        row3[2]=ac[(C,C)]
        tuple(row3)
        bd:Board=(row1,row2,row3)
        return bd

def createAccess(bd:Board)->Access:
        ac:Access=dict()
        ac[(A,A)]=bd[0][0]
        ac[(A,B)]=bd[0][1]
        ac[(A,C)]=bd[0][2]
        ac[(B,A)]=bd[1][0]
        ac[(B,B)]=bd[1][1]
        ac[(B,C)]=bd[1][2]
        ac[(C,A)]=bd[2][0]
        ac[(C,B)]=bd[2][1]
        ac[(C,C)]=bd[2][2]
        return ac
```

The Creating the Board from the Access and Access from the Board is required as at some places we have to use Access to get elements of the board or to make changes to board and as the board is the tuple(immutable) new board can be created using the above method from access.

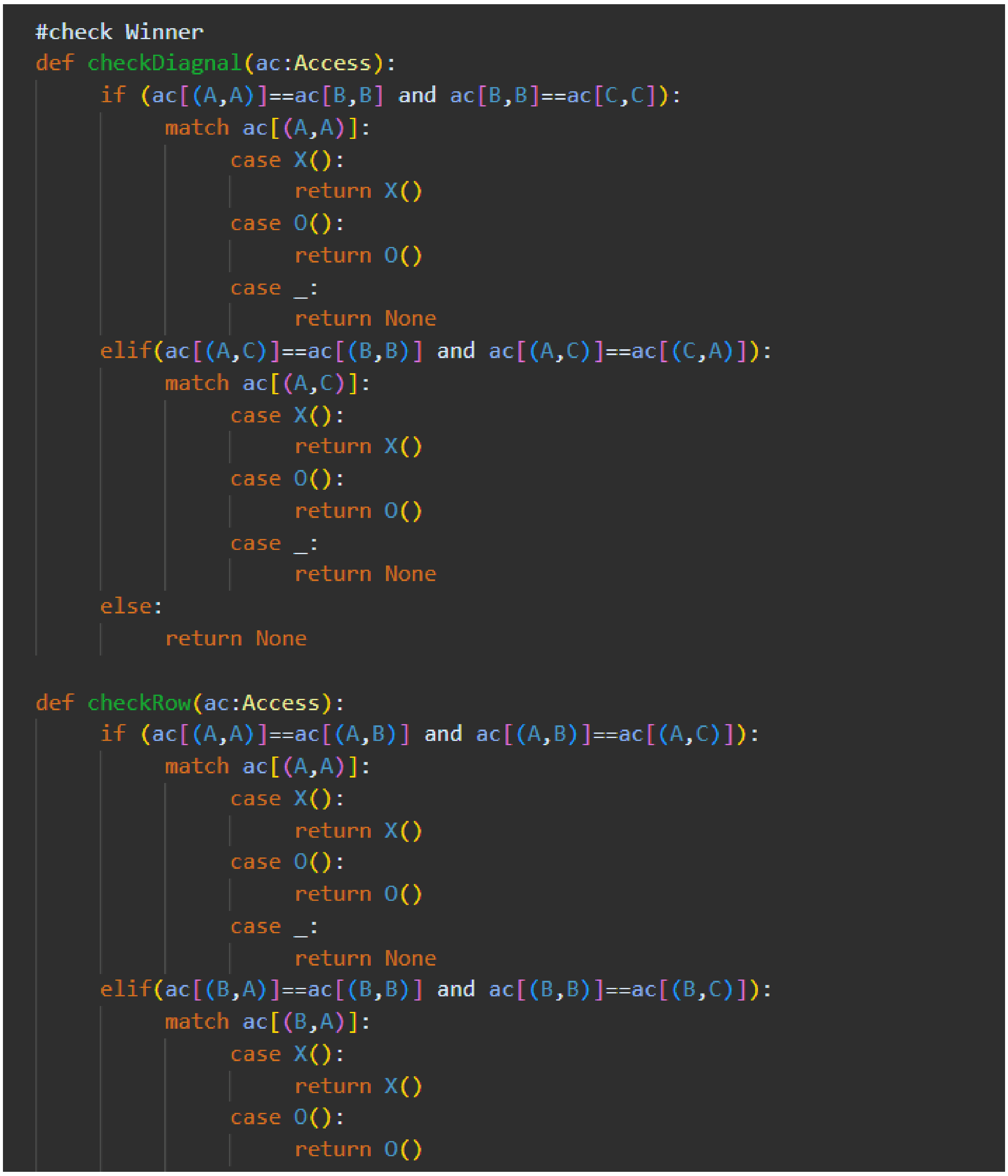
Playing Token on the Board:



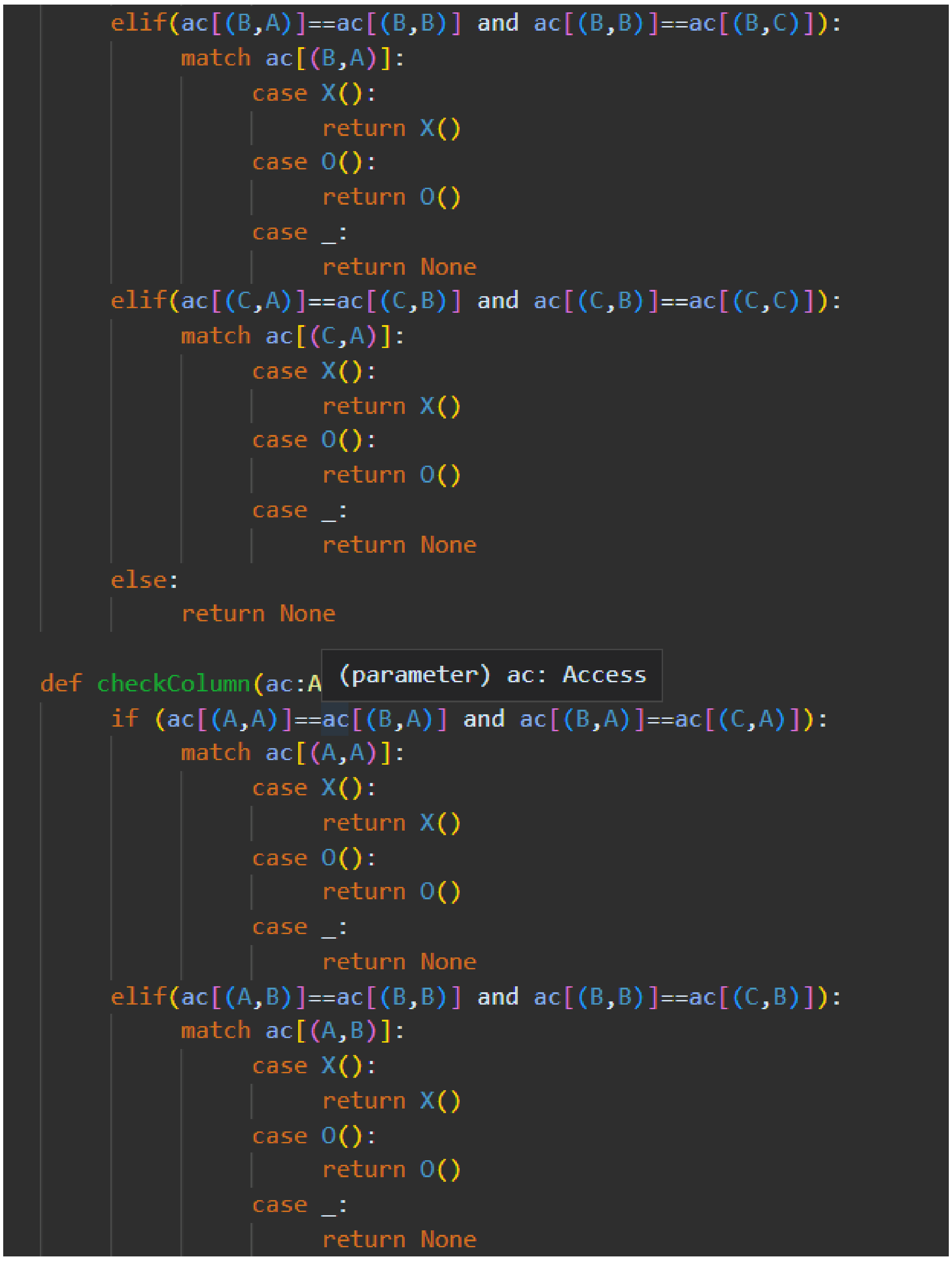
```
# play Token
def playToken(tk:Token,bd:Board)->Board:
    ac:Access=createAccess(bd)
    cord:Cord=getCoordinates()
    if (ac[cord]== X() or ac[cord]==O()):
            print("Token Already Placed:Choose Another Spot")
            return bd
    else:
            ac[cord]=tk
            bd=createBoard(ac)
            return bd

def playTokenByBot(tk:Token,bd:Board)->Board:
    access:Access=createAccess(bd)
    axiss=(A,B,C)

    for row in axiss:
        for column in axiss:
            if (access[(row,column)]==None):
                access[(row,column)]=tk
                return createBoard(access)
```

The Token can be played on the board using the playToken method. The play token method takes in Token and Board as the input and ask for the coordinates using the getCoordinate method and plays the token at given coordinate and returns the new updated board by creating it from the access.

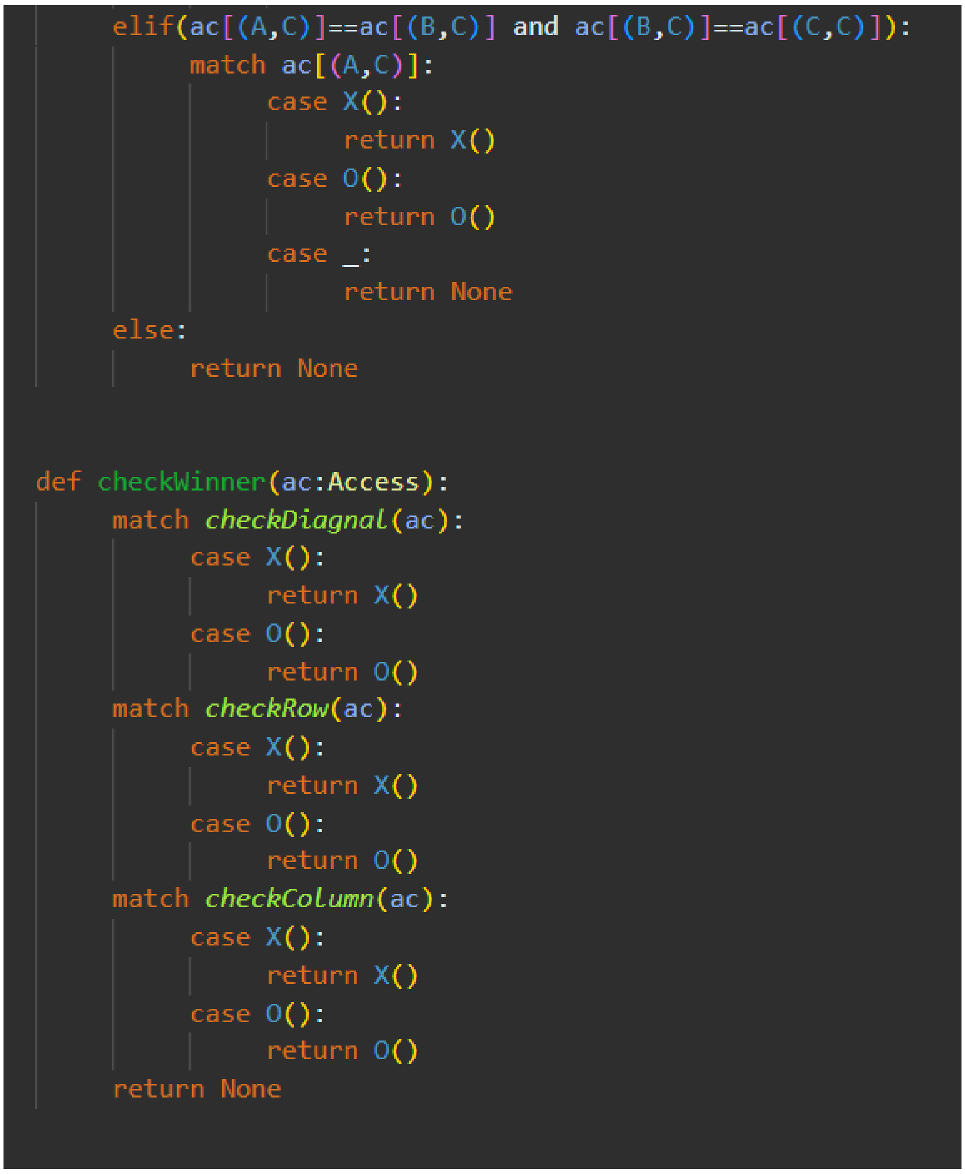
Checking Winner:



```python
#check Winner
def checkDiagnal(ac:Access):
    if (ac[(A,A)]==ac[B,B] and ac[B,B]==ac[C,C]):
        match ac[(A,A)]:
            case X():
                return X()
            case O():
                return O()
            case _:
                return None
    elif(ac[(A,C)]==ac[(B,B)] and ac[(A,C)]==ac[(C,A)]):
        match ac[(A,C)]:
            case X():
                return X()
            case O():
                return O()
            case _:
                return None
    else:
        return None

def checkRow(ac:Access):
    if (ac[(A,A)]==ac[(A,B)] and ac[(A,B)]==ac[(A,C)]):
        match ac[(A,A)]:
            case X():
                return X()
            case O():
                return O()
            case _:
                return None
    elif(ac[(B,A)]==ac[(B,B)] and ac[(B,B)]==ac[(B,C)]):
        match ac[(B,A)]:
            case X():
                return X()
            case O():
                return O()
```



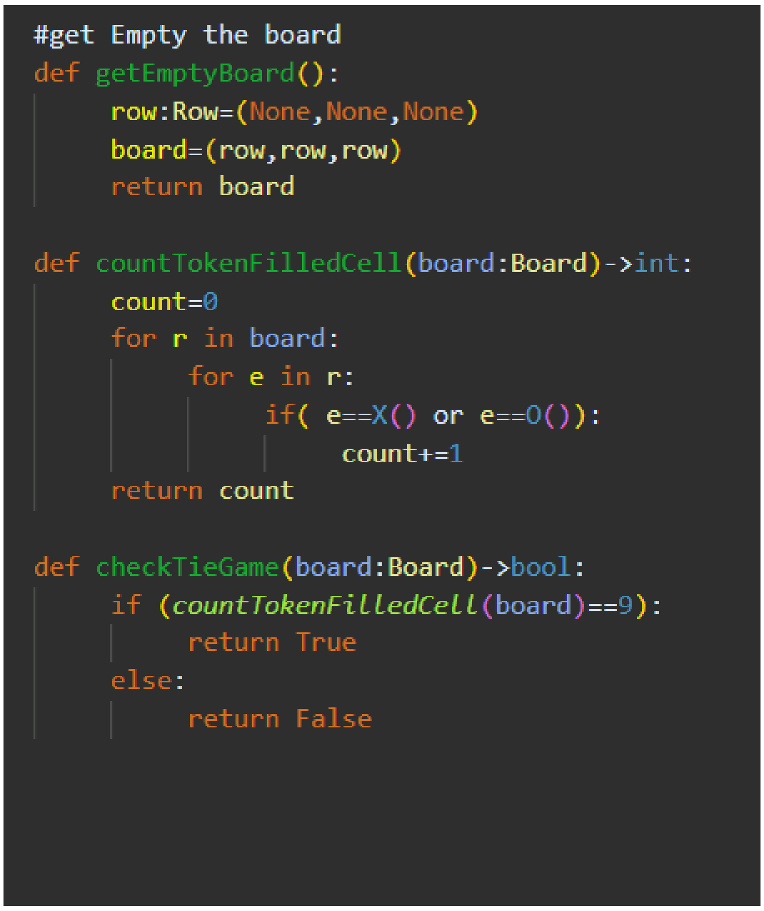
```
    elif(ac[(B,A)]==ac[(B,B)] and ac[(B,B)]==ac[(B,C)]):
        match ac[(B,A)]:
            case X():
                return X()
            case O():
                return O()
            case _:
                return None
    elif(ac[(C,A)]==ac[(C,B)] and ac[(C,B)]==ac[(C,C)]):
        match ac[(C,A)]:
            case X():
                return X()
            case O():
                return O()
            case _:
                return None
    else:
        return None

def checkColumn(ac:A (parameter) ac: Access
    if (ac[(A,A)]==ac[(B,A)] and ac[(B,A)]==ac[(C,A)]):
        match ac[(A,A)]:
            case X():
                return X()
            case O():
                return O()
            case _:
                return None
    elif(ac[(A,B)]==ac[(B,B)] and ac[(B,B)]==ac[(C,B)]):
        match ac[(A,B)]:
            case X():
                return X()
            case O():
                return O()
            case _:
                return None
```



```
    elif(ac[(A,C)]==ac[(B,C)] and ac[(B,C)]==ac[(C,C)]):
        match ac[(A,C)]:
            case X():
                return X()
            case O():
                return O()
            case _:
                return None
    else:
        return None


def checkWinner(ac:Access):
    match checkDiagnal(ac):
        case X():
            return X()
        case O():
            return O()
    match checkRow(ac):
        case X():
            return X()
        case O():
            return O()
    match checkColumn(ac):
        case X():
            return X()
        case O():
            return O()
    return None
```

Using the Access the diagonals, row and column of the board are checked. after checking the board the check winner class returns the token of the winning player.

Counting Board and Checking match Tied:



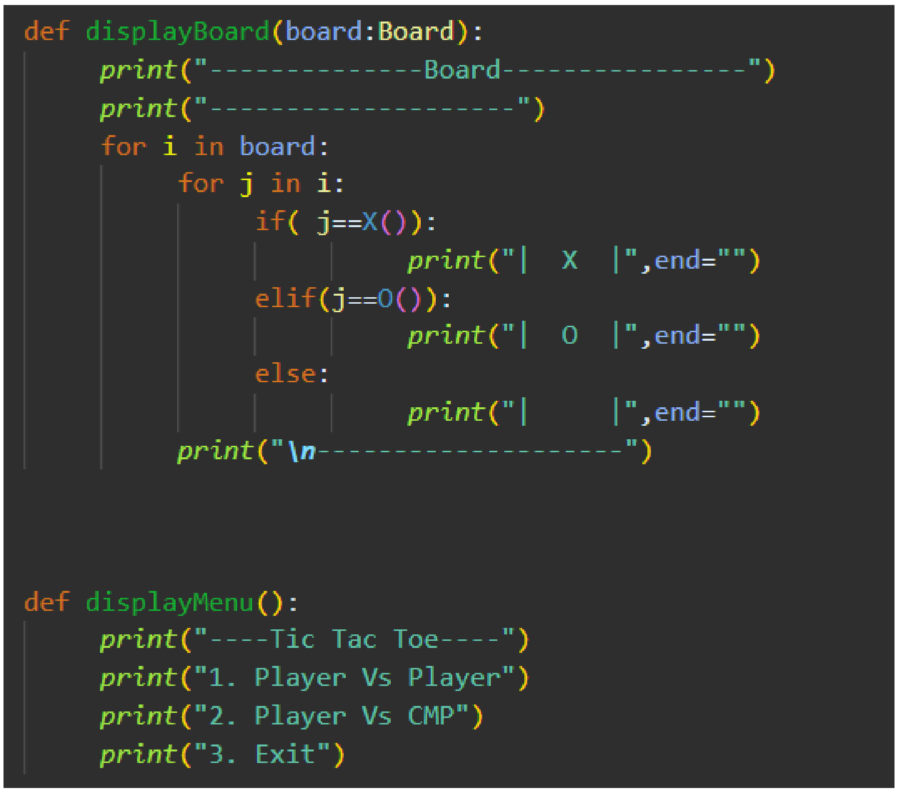
```python
#get Empty the board
def getEmptyBoard():
    row:Row=(None,None,None)
    board=(row,row,row)
    return board

def countTokenFilledCell(board:Board)->int:
    count=0
    for r in board:
        for e in r:
            if( e==X() or e==O()):
                count+=1
    return count

def checkTieGame(board:Board)->bool:
    if (countTokenFilledCell(board)==9):
        return True
    else:
        return False
```

getEmptyBoard method returns the empty board which can be used while restarting the game

Count token Filled cell checks out how many cells are been filled in the Token and return number count

Check Tie game checks if the board is full and still there is no winner and game is considered to be a tie and Boolean is return by the method True if Tie and False if not.

Display Board and Menu:

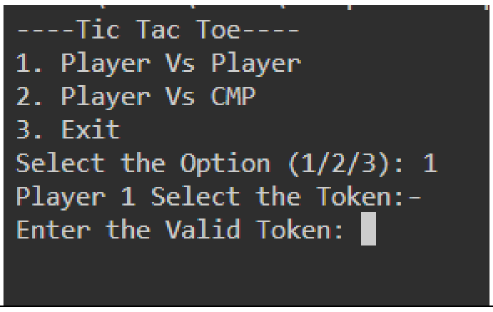


```
def displayBoard(board:Board):
    print("--------------Board----------------")
    print("--------------------")
    for i in board:
        for j in i:
            if( j==X()):
                    print("|  X  |",end="")
            elif(j==O()):
                    print("|  O  |",end="")
            else:
                    print("|     |",end="")
        print("\n--------------------")



def displayMenu():
    print("----Tic Tac Toe----")
    print("1. Player Vs Player")
    print("2. Player Vs CMP")
    print("3. Exit")
```

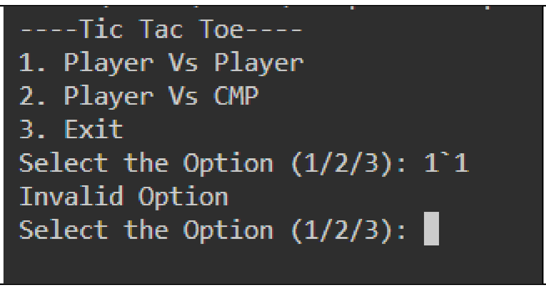
Display board and Menu are used during the user interaction to display the menu to select option and board to play next move using this method these methods do not take /give any input and output.

Testing:

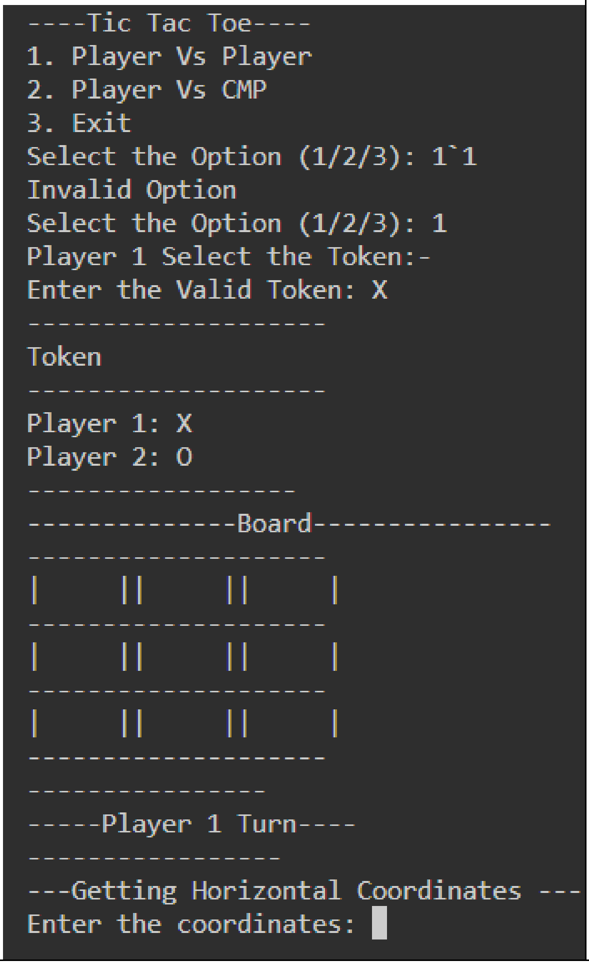
TC1	Input	Expected Output	Actual Output	Passed/Failed

Checking For Valid Option Selected at the starting display menu	1.Run the Program
	2.Asks for the input after displaying the Menu
	3.”1”option Selected
		Run Properly and ask Player 1 for the Token	Program worked properly and asked for the Token
			
			`----Tic Tac Toe----`
			`1. Player Vs Player`
			`2. Player Vs CMP`
			`3. Exit`
			`Select the Option (1/2/3): 1`
			`Player 1 Select the Token:-`
			`Enter the Valid Token:`
				Passed

TC2	Input	Expected Output	Actual Output	Passed/Failed
Checking for the invalid option selected	1. Run Program	Give out the Invalid Option Messag	Given out the invalid option message and asked for the input	Passed

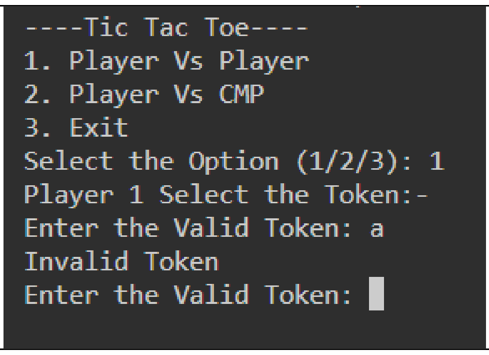
start of the display menu	2. Ask For the Input 3. Invalid option Selected	e and ask for the input Again		

```
----Tic Tac Toe----
1. Player Vs Player
2. Player Vs CMP
3. Exit
Select the Option (1/2/3): 1`1
Invalid Option
Select the Option (1/2/3): 
```

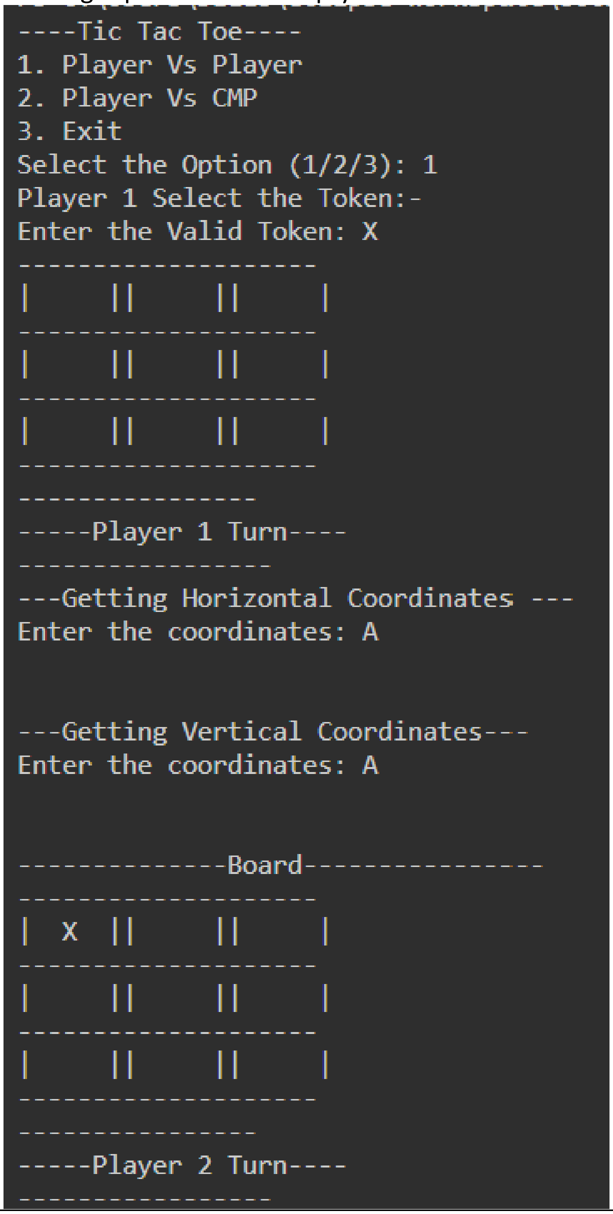
TC3	Input	Expected Output	Actual Output	Passed/Failed
selecting the Valid Token	1.Run the Program 2.Asks for the input after displaying the Menu 3.”1”option Selected 4.Ask For the selection of the Token 5. give a Valid token	Ask the player to select the Coordinates and the Game begin with displaying the token and the board.	Displayed Tokens For both players with the board and asked for the coordinates. 	passed

```
----Tic Tac Toe----
1. Player Vs Player
2. Player Vs CMP
3. Exit
Select the Option (1/2/3): 1`1
Invalid Option
Select the Option (1/2/3): 1
Player 1 Select the Token:-
Enter the Valid Token: X
--------------------
Token
--------------------
Player 1: X
Player 2: O
------------------
--------------Board----------------
--------------------
|     ||     ||     |
--------------------
|     ||     ||     |
--------------------
|     ||     ||     |
--------------------
----------------
-----Player 1 Turn----
----------------
---Getting Horizontal Coordinates ---
Enter the coordinates: 
```

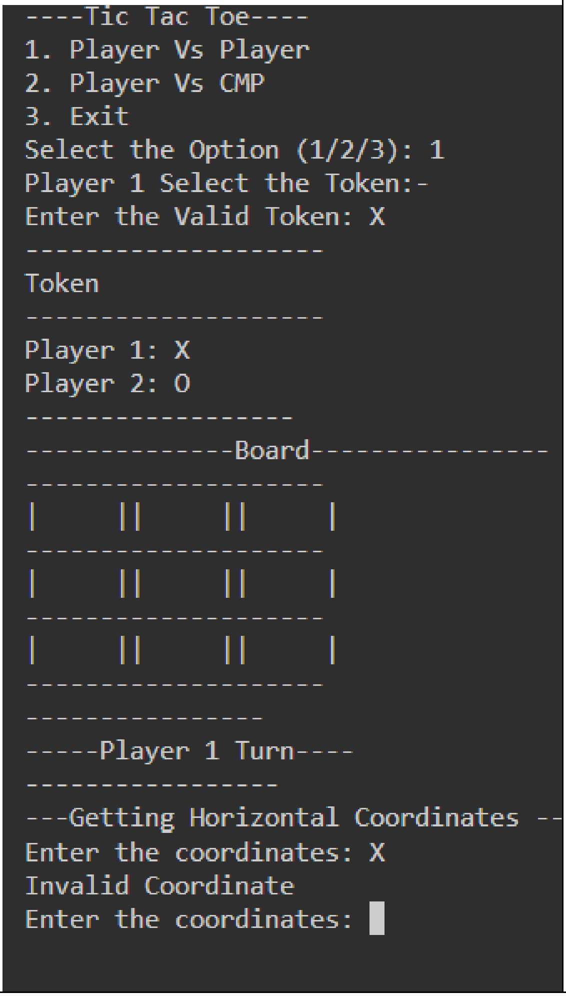
TC4	Input	Expected Output	Actual Output	Passed/Failed
Selecting the invalid Token	1.Run the Program 2.Asks for the input after displaying the Menu	Display invalid Token Message and ask for the token again	Displayed invalid Token message and asked for the token again	passed

	3.”1”option Selected 4.Ask For the selection of the Token 5. give an invalid token			

```
----Tic Tac Toe----
1. Player Vs Player
2. Player Vs CMP
3. Exit
Select the Option (1/2/3): 1
Player 1 Select the Token:-
Enter the Valid Token: a
Invalid Token
Enter the Valid Token:
```

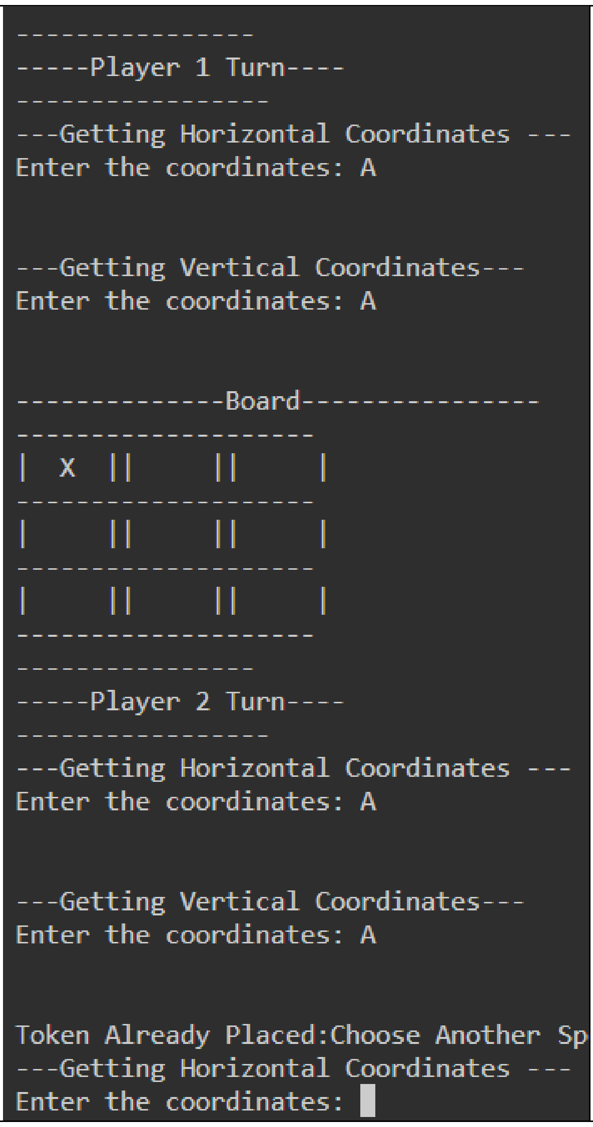
TC5	Input	Expected Output	Actual Output	Passed / Failed
Select a valid Coordinates and cell is empty	1.Run the Program 2.Asks for the input after displaying the Menu 3.”1”option Selected 4.Ask For the selection of the Token 5. give a Valid token 6. ask for the coordinates from the first player 7. give a valid coordinates of the empty cell	Token get placed of the empty space can be seen on the displayed board	Token got placed at the empty cell 	Passed

```
----Tic Tac Toe----
1. Player Vs Player
2. Player Vs CMP
3. Exit
Select the Option (1/2/3): 1
Player 1 Select the Token:-
Enter the Valid Token: X
--------------------
|    ||    ||    |
--------------------
|    ||    ||    |
--------------------
|    ||    ||    |
--------------------
----------------
-----Player 1 Turn----
----------------
---Getting Horizontal Coordinates ---
Enter the coordinates: A


---Getting Vertical Coordinates---
Enter the coordinates: A


--------------Board----------------
--------------------
|  X ||    ||    |
--------------------
|    ||    ||    |
--------------------
|    ||    ||    |
--------------------
----------------
-----Player 2 Turn----
----------------
```

TC7	Input	Expected Output	Actual Output	Passed/Failed
Select an invalid Coordinates	1.Run the Program 2.Asks for the input after displaying the Menu 3.”1”option Selected 4.Ask For the selection of the Token 5. give a Valid token 6. ask for the coordinates from the first player 7. give an invalid coordinate of the empty cell	Display invalid coordinates message and ask for the coordinates again	Displayed invalid coordinates message and asked for the coordinates 	Passed

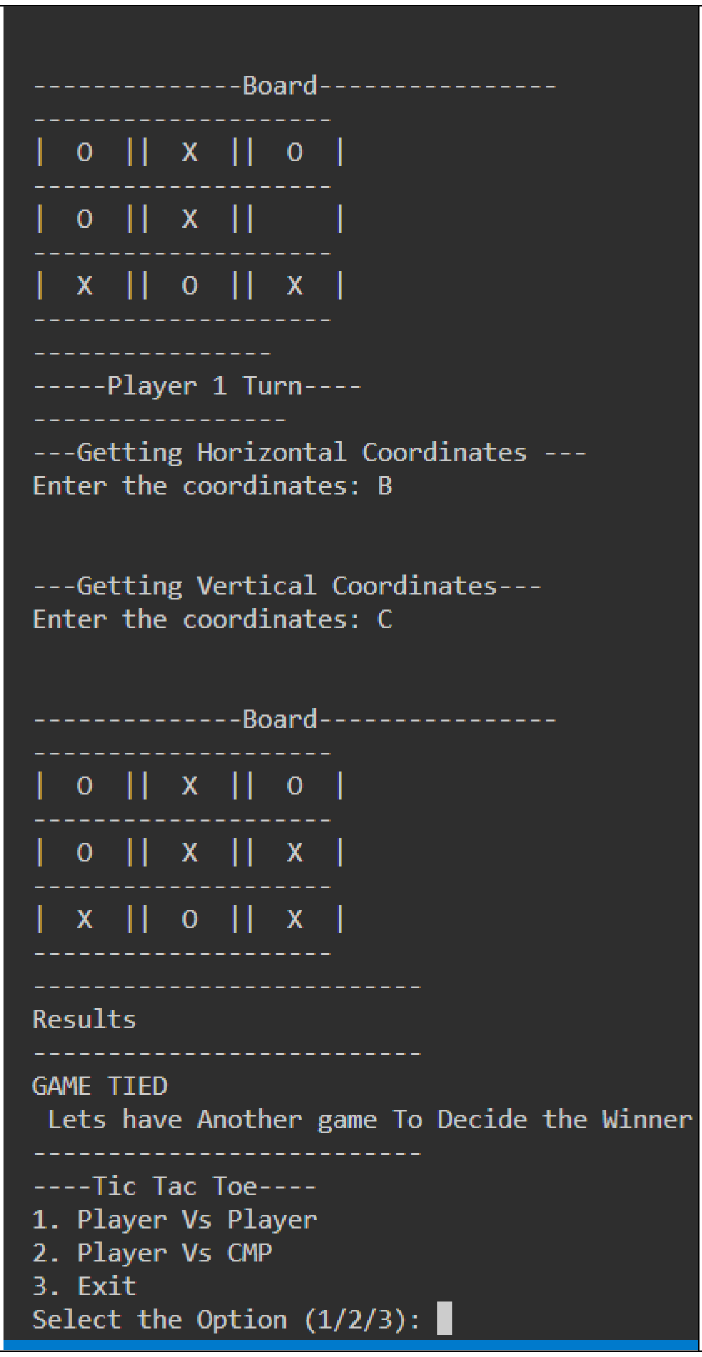
```
----Tic Tac Toe----
1. Player Vs Player
2. Player Vs CMP
3. Exit
Select the Option (1/2/3): 1
Player 1 Select the Token:-
Enter the Valid Token: X
--------------------
Token
--------------------
Player 1: X
Player 2: O
------------------
--------------Board----------------
--------------------
|     ||     ||     |
--------------------
|     ||     ||     |
--------------------
|     ||     ||     |
--------------------
----------------
-----Player 1 Turn----
-----------------
---Getting Horizontal Coordinates --
Enter the coordinates: X
Invalid Coordinate
Enter the coordinates: 
```

TC8	Input	Expected Output	Actual Output	Passed/Failed
Select a valid coordinate but the cell is filled	1.Run the Program 2.Asks for the input after displaying the Menu 3.”1”option Selected 4.Ask For the selection of the Token 5. give a Valid token	Display spot is already filled message and ask for the coordinates again	Displayed the message and asked for the coordinates	Passed

	6. ask for the coordinates from the first player 7. give a valid coordinate of the filled cell			

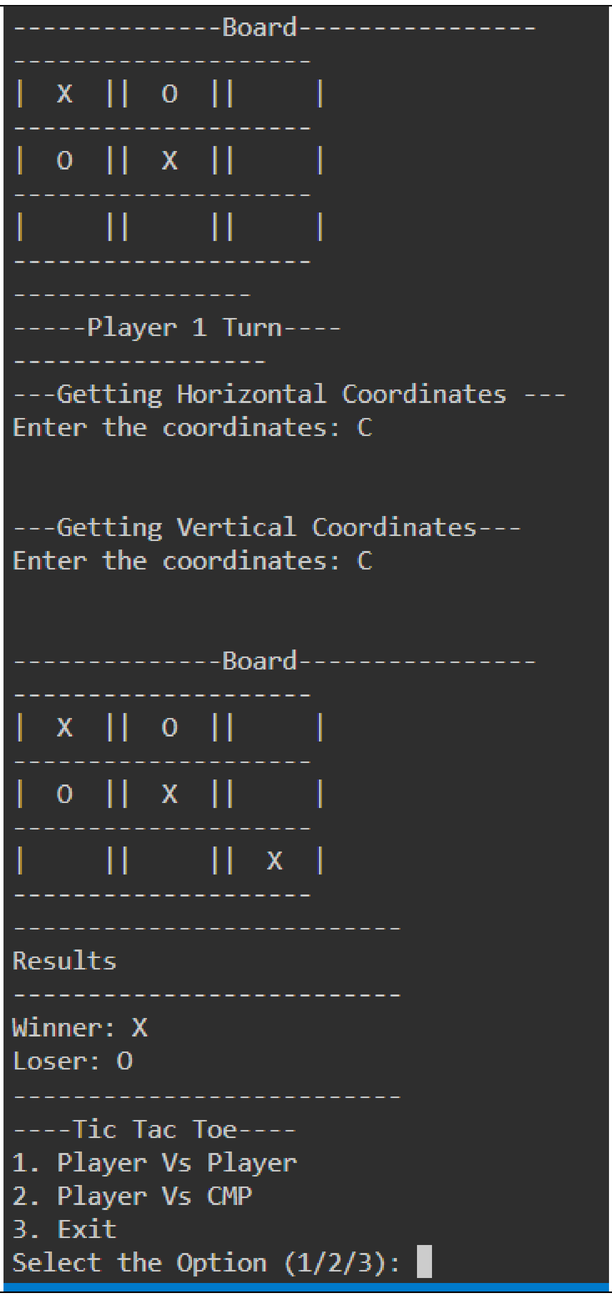
```
----------------
-----Player 1 Turn----
-----------------
---Getting Horizontal Coordinates ---
Enter the coordinates: A


---Getting Vertical Coordinates---
Enter the coordinates: A


--------------Board----------------
--------------------
|  X  ||     ||     |
--------------------
|     ||     ||     |
--------------------
|     ||     ||     |
--------------------
----------------
-----Player 2 Turn----
-----------------
---Getting Horizontal Coordinates ---
Enter the coordinates: A


---Getting Vertical Coordinates---
Enter the coordinates: A


Token Already Placed:Choose Another Sp
---Getting Horizontal Coordinates ---
Enter the coordinates: 
```

TC9	Input	Expected Output	Actual Output	Passed/Failed
Check the match is tied	1.Run the Program 2.Asks for the input after displaying the Menu 3.”1”option Selected 4.Ask For the selection	Gives a message that game has been tied and ask for the option again	Given Tied game message and ask for the option again	Passed

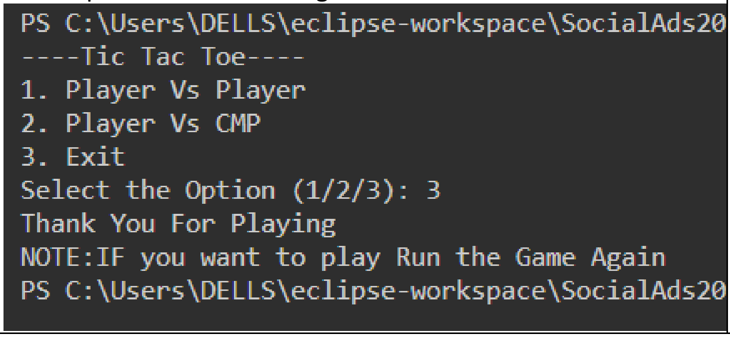
	of the Token 5. give a Valid token 6. ask for the coordinates from the first player 7. give a valid coordinate for both the player let no one win and let the board fully filled			

```
--------------Board----------------
--------------------
| O || X || O |
--------------------
| O || X ||   |
--------------------
| X || O || X |
--------------------
----------------
-----Player 1 Turn----
----------------
---Getting Horizontal Coordinates ---
Enter the coordinates: B


---Getting Vertical Coordinates---
Enter the coordinates: C


--------------Board----------------
--------------------
| O || X || O |
--------------------
| O || X || X |
--------------------
| X || O || X |
--------------------
--------------------------
Results
--------------------------
GAME TIED
 Lets have Another game To Decide the Winner
--------------------------
----Tic Tac Toe----
1. Player Vs Player
2. Player Vs CMP
3. Exit
Select the Option (1/2/3):
```

TC10	Input	Expected Output	Actual Output	Passed/Failed
Check if Token won the match	1.Run the Program 2.Asks for the input after displaying the Menu 3.”1”option Selected 4.Ask For the	Display the result message and ask for the option selection again	Displayed the result message and asked for the selected message	Passed

	selection of the Token 5. give a Valid token 6. ask for the coordinates from the first player 7. give a valid coordinate of the empty cell and make one player win			

```
--------------Board----------------
--------------------
|  X  ||  O  ||     |
--------------------
|  O  ||  X  ||     |
--------------------
|     ||     ||     |
--------------------
----------------
-----Player 1 Turn----
-----------------
---Getting Horizontal Coordinates ---
Enter the coordinates: C


---Getting Vertical Coordinates---
Enter the coordinates: C


--------------Board----------------
--------------------
|  X  ||  O  ||     |
--------------------
|  O  ||  X  ||     |
--------------------
|     ||     ||  X  |
--------------------
--------------------------
Results
--------------------------
Winner: X
Loser: O
--------------------------
----Tic Tac Toe----
1. Player Vs Player
2. Player Vs CMP
3. Exit
Select the Option (1/2/3): 
```

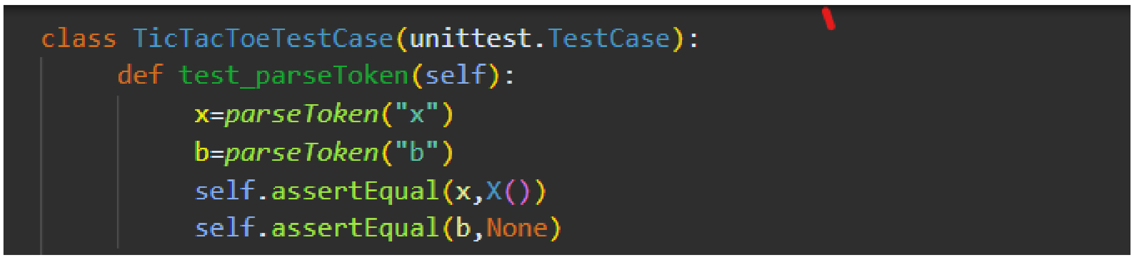
TC11	Input	Expected Output	Actual Output	Passed/Failed
Quit	1.Run the Program 2.Asks for the input after displaying the Menu	Game quitted with good bye message	Game quitted with a message 	Passed

```
PS C:\Users\DELLS\eclipse-workspace\SocialAds20
----Tic Tac Toe----
1. Player Vs Player
2. Player Vs CMP
3. Exit
Select the Option (1/2/3): 3
Thank You For Playing
NOTE:IF you want to play Run the Game Again
PS C:\Users\DELLS\eclipse-workspace\SocialAds20
```

	3.”3”option Selected			

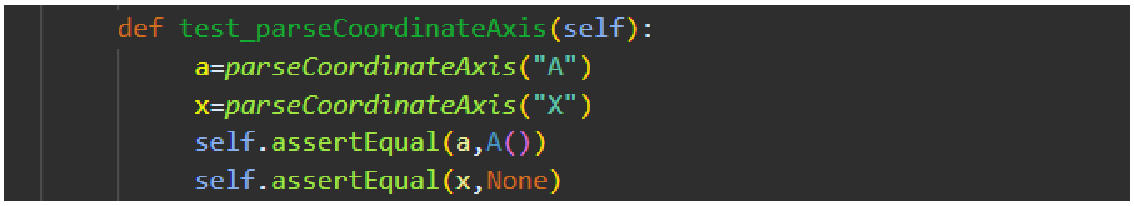
Unit Testing

This Functions which are tested in the Unit testing are building blocks of the game so the Testing of this function ensures the stability and functionality of the program.



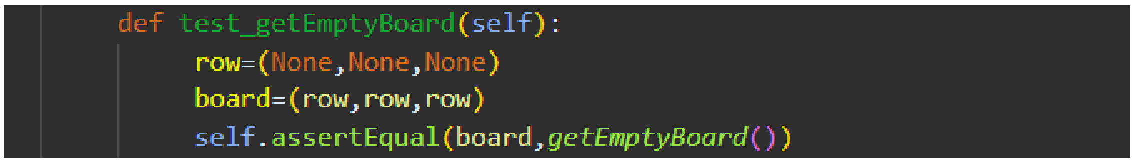
```
class TicTacToeTestCase(unittest.TestCase):
    def test_parseToken(self):
        x=parseToken("x")
        b=parseToken("b")
        self.assertEqual(x,X())
        self.assertEqual(b,None)
```

1. This test checks if the parseToken method return the correct Token . x and b variables are storing parse Token results to string “x” and “b” is compared to the expected output results which is X() and None(if invalid Token like “b”).



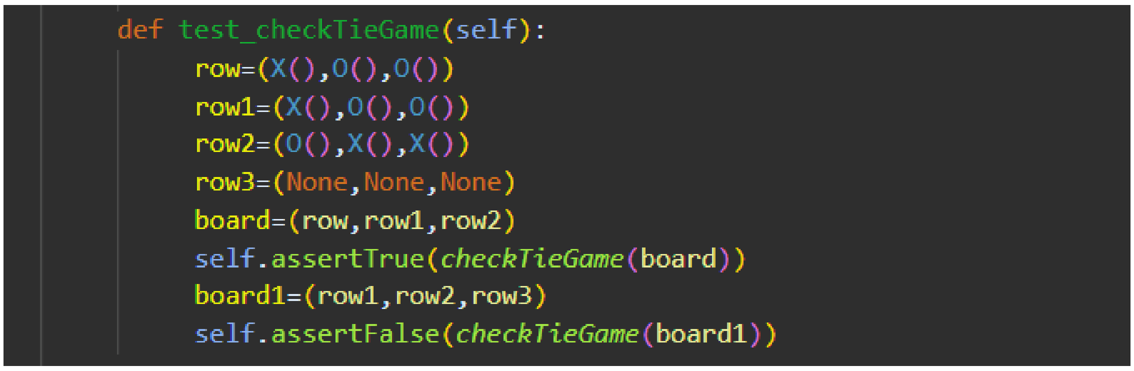
```
    def test_parseCoordinateAxis(self):
        a=parseCoordinateAxis("A")
        x=parseCoordinateAxis("X")
        self.assertEqual(a,A())
        self.assertEqual(x,None)
```

2. This Test checks if the parseCoordinateAxis Function return the correct coordinates. a,x variables are Storing parse coordinates result and then compared with expected result if invalid Axis like “X” than return None



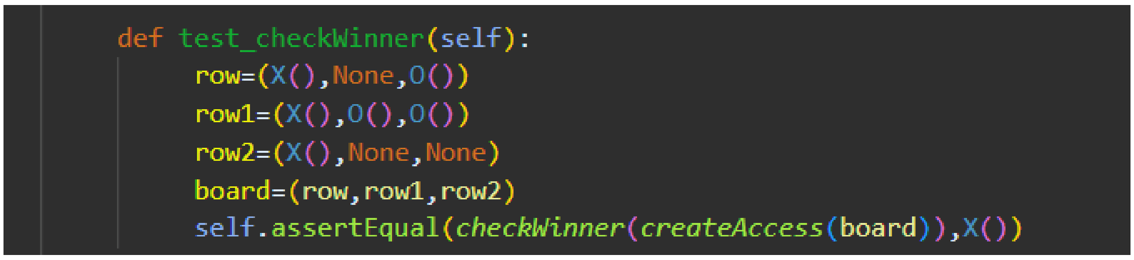
```
    def test_getEmptyBoard(self):
        row=(None,None,None)
        board=(row,row,row)
        self.assertEqual(board,getEmptyBoard())
```

3. This Test checks if the board returned by getEmptyBoard () is really Empty so board variable Stores an empty board which is compared with getEmptyBoard function returned board.



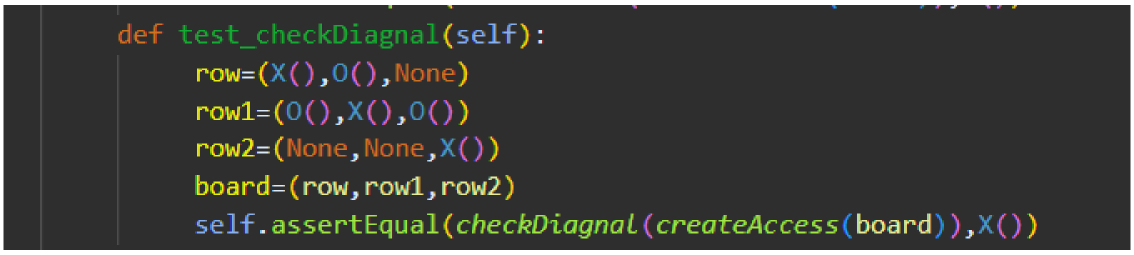
```python
def test_checkTieGame(self):
    row=(X(),O(),O())
    row1=(X(),O(),O())
    row2=(O(),X(),X())
    row3=(None,None,None)
    board=(row,row1,row2)
    self.assertTrue(checkTieGame(board))
    board1=(row1,row2,row3)
    self.assertFalse(checkTieGame(board1))
```

4. This test Checks if the checkTieGame returns the correct Boolean for the game is finished and tied than True or game is going on than False.



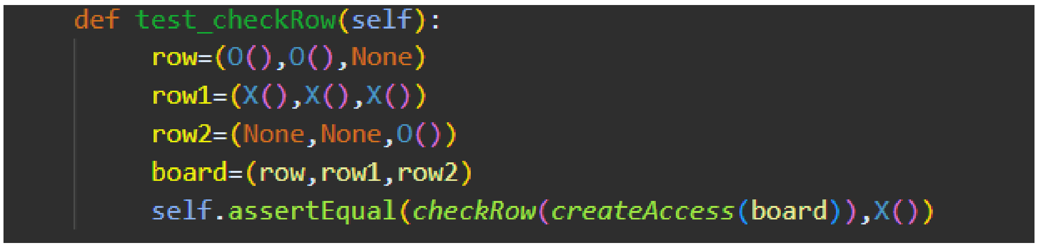
```python
def test_checkWinner(self):
    row=(X(),None,O())
    row1=(X(),O(),O())
    row2=(X(),None,None)
    board=(row,row1,row2)
    self.assertEqual(checkWinner(createAccess(board)),X())
```

5. This test checks if the Winner is Found by the game when board is given where X is the winner and should return the winner Token which will be X



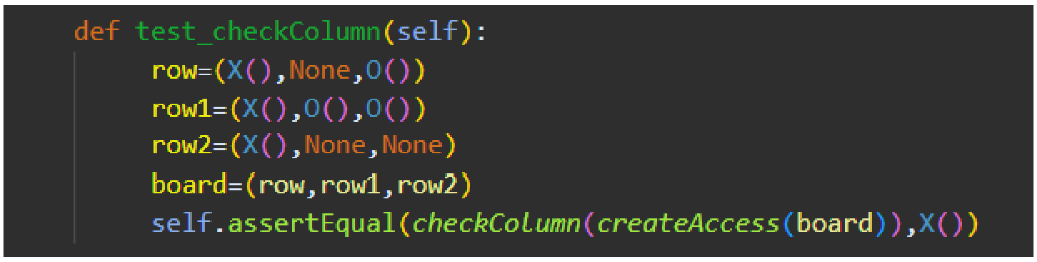
```python
def test_checkDiagnal(self):
    row=(X(),O(),None)
    row1=(O(),X(),O())
    row2=(None,None,X())
    board=(row,row1,row2)
    self.assertEqual(checkDiagnal(createAccess(board)),X())
```

6. This Test Checks if the Token win the game by creating Diagonally should return the winner Token which is X and is compared with Expected result X.



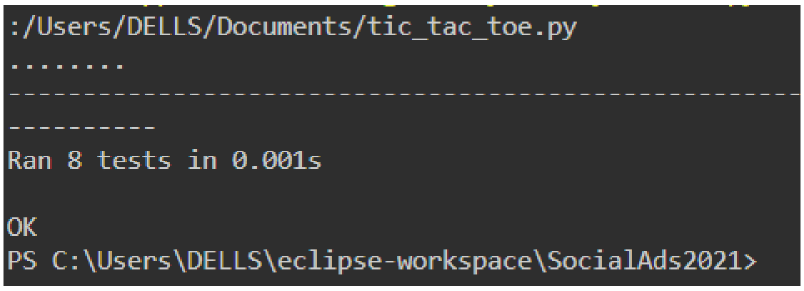
```python
def test_checkRow(self):
    row=(O(),O(),None)
    row1=(X(),X(),X())
    row2=(None,None,O())
    board=(row,row1,row2)
    self.assertEqual(checkRow(createAccess(board)),X())
```

7. This test Checks if the Token win by placing three consecutive Token in Row and return the found token with the same.



```
    def test_checkColumn(self):
        row=(X(),None,O())
        row1=(X(),O(),O())
        row2=(X(),None,None)
        board=(row,row1,row2)
        self.assertEqual(checkColumn(createAccess(board)),X())
```

8. This test check if the Token win by placing three consecutive Token in column and return the Token with the Same



```
:/Users/DELLS/Documents/tic_tac_toe.py
........
----------------------------------------------------------------------
Ran 8 tests in 0.001s

OK
PS C:\Users\DELLS\eclipse-workspace\SocialAds2021>
```

Running the test Showed that the above test has been passed with no error. This show that the functions created are working Expectedly and are handling the error and invalid inputs.

Documentation for Contributors:

Documentation consists of understanding of the programming basic codes, planning of the project, expected output of the project and codebase of the older version.

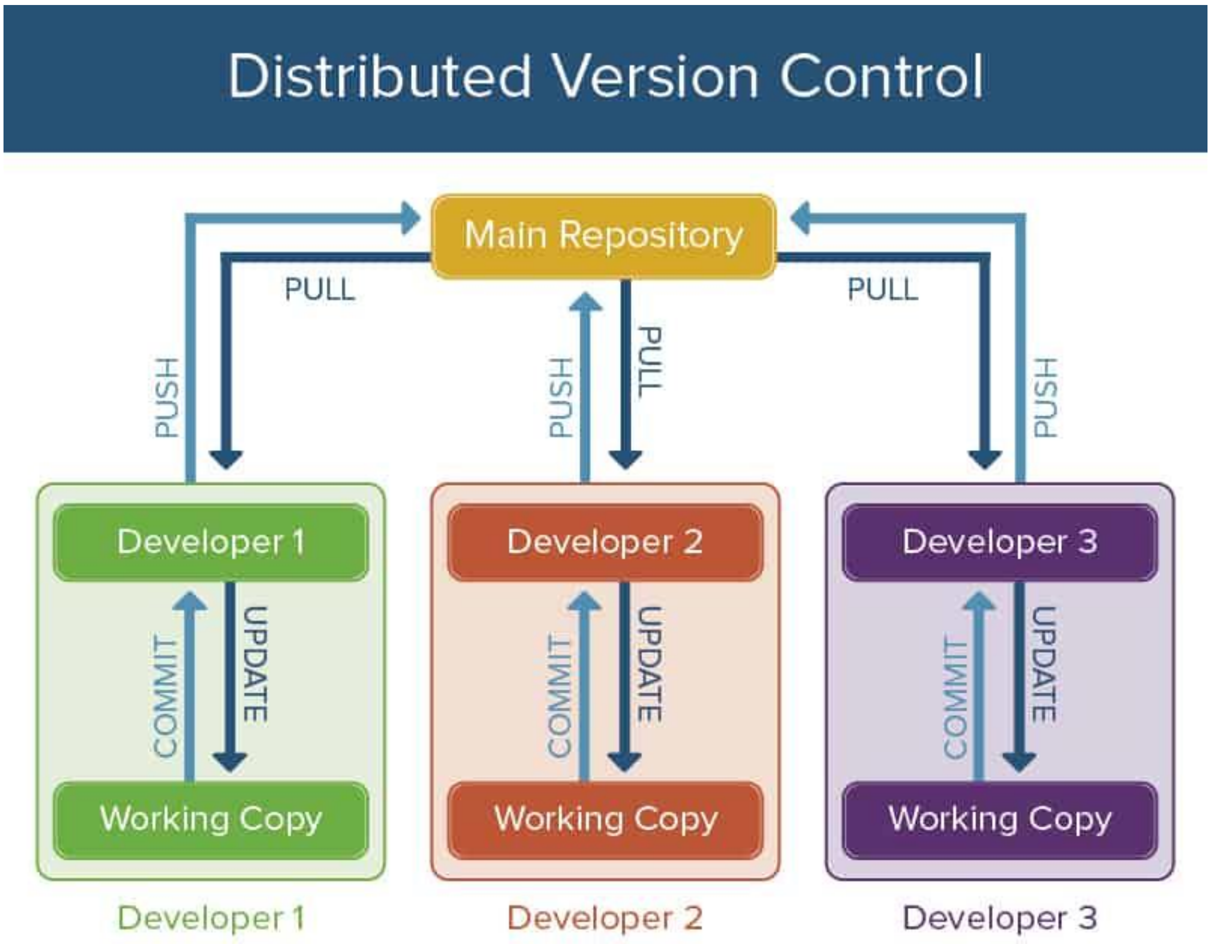
Providing the Documentation to the contributors can help in boost up the speed of the project by onboarding new contributors which will can also quickly grasp the project basic codes and become familiar with the project. I also led to improve the understanding of the new contributors about the project and make them interact with the project more efficiently.

By all the contributors having same documentation it improves their collaboration and by making them understand each other‘s work.

Version Control:

Version control is also known as Source control, I practice of tracking managing changes to software code .Version Control systems ,such as Git, Play an important role in managing the software project with multiple contributors.

Version control like git provide a centralised repository for storing and tracking the changes.



Push

Push is used to transfer the local working copy changes to the main repository. This allows you to share your Local commits with other contributors through the repository.

Pull

Pull command is used to fetch the changes from the main repository to your local repository copy.

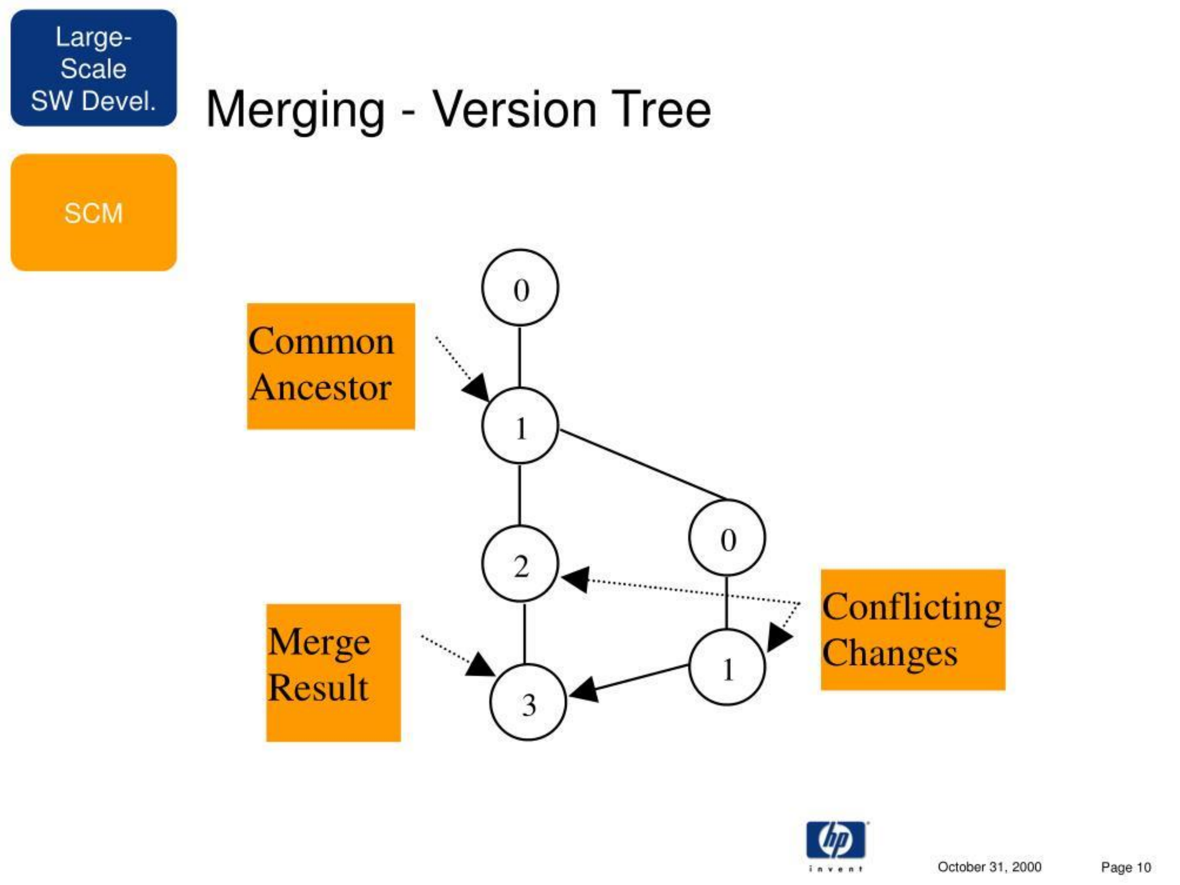
Commit

Commit operation is used to keep the track of the history of the version of the code and modification made in file or repository directories. It captures the snapshot of the current state of the codebase.

Update:

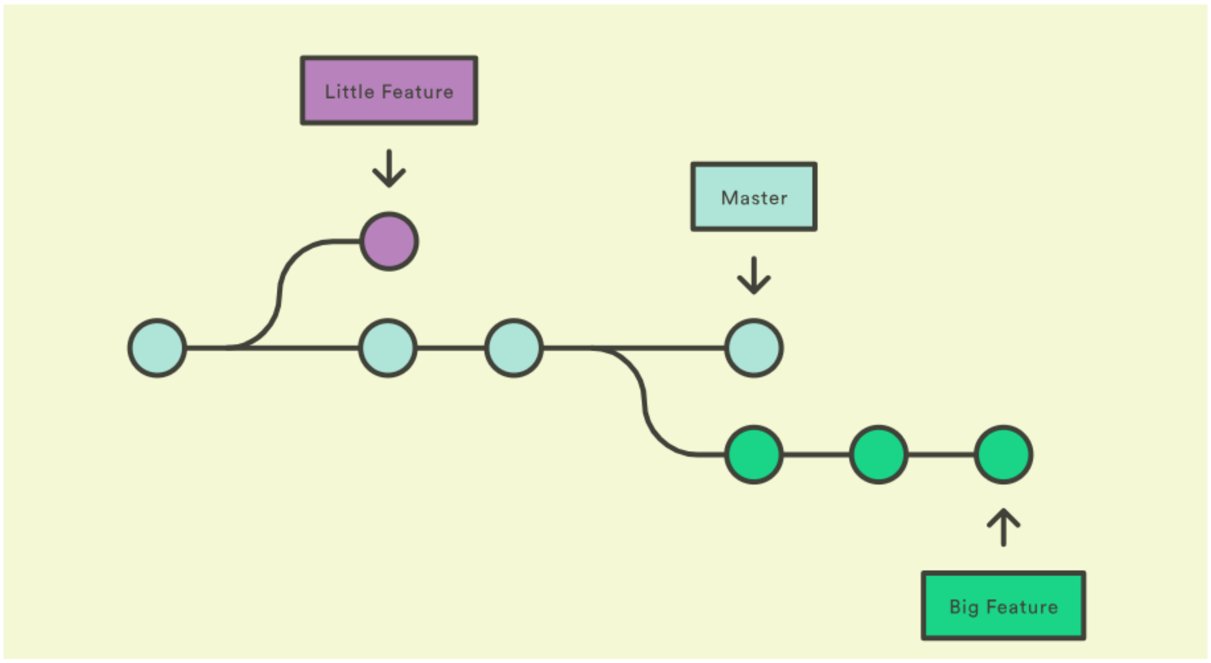
Update Operation is used to get the repository content to update the feature or coding which then can be commit to the repository which can then be the new version of that content and history of that content will be saved by the commit.

Merge



Merge operation is used to combine changes from different branches or copy into single branch. Allows to merge changes of one branch to another.

Branch



A branch in the version control is the separate line of development diverging from the main line of the development it helps in different collaborators to work in different features in the software and merge them into the main branch after completion. Branching enables software development team to work on different parts of the project without impacting each other.

Benefits of version control

Benefit	Explanation
Branching	Version control system has feature called branching allowing contributors to work on different feature or bug fixes of the project independently making it easier to manage and review changes before merging them into main branch.
Tracking history	Version control system keep a detailed history of all changes to the repository. Changes is recorded as the commit. Which help in understanding the evolution of project by differentiating between the versions.
Merging	Merging of two codes can be done in the version control system. When there is conflict in codes like when two contributors modify the same file there can be some conflict in code. Which can be solved by merging them by comparing and making adjustment in controlled manner.
Collaboration	The version control system let the contributors to make a working copy of the same project and make changes independently and merge the work after the changes which enables real time collaboration.

Leverage the version control in developing software projects

By leveraging the version control I can keep track of the previous versions of my project using commit history and helps me to monitor the improvement of the project which will be excellent learning practice.

Version control is great help in working the group as it improves the real life collaboration ,get less chances of bugs by manually merging two same feature codes by two group members. Version control also help in boosting the speed of work as different members can work in different features of project using branching also keeping track of other members through commit and merging the work into one more efficiently and in controlled manner without bugs and errors.

References

Distributed version Control image: The Basics of Software Version Control | Smartsheet

What is Version Control?: What is version control | Atlassian Git Tutorial

Benefits of Version Control: What is version control | Atlassian Git Tutorial

Branch image Example: Github Branching And Version Control Explained - Konstantinfo

What is merging?: Merge (version control) - Wikipedia

What is branching?: What Is a Branch? | Perforce