Step 1: Write a function that can print out a board. Set up your board as a list, where each index 1-9 corresponds with a number on a number pad, so you get a 3 by 3 board representation.

In [1]:

```
from IPython.display import clear_output

def display_board(board):
    clear_output()  # Remember, this only works in jupyter!

    print('   |   |')
    print(' ' + board[7] + ' | ' + board[8] + ' | ' + board[9])
    print('   |   |')
    print('-----------')
    print('   |   |')
    print(' ' + board[4] + ' | ' + board[5] + ' | ' + board[6])
    print('   |   |')
    print('-----------')
    print('   |   |')
    print(' ' + board[1] + ' | ' + board[2] + ' | ' + board[3])
    print('   |   |')
```

In [2]:

```
test_board = ['#','X','0','X','0', 'X', '0', 'X', '0','X']
display_board(test_board)
```

```
   |   |
 X | 0 | X
   |   |
-----------
   |   |
 0 | X | 0
   |   |
-----------
   |   |
 X | 0 | X
   |   |
```

Step 2: Write a function that can take in a player input and assign their marker as 'X' or 'O'. Think about using while loops to continually ask until you get a correct answer.

In [3]:

```
def player_input():
    marker = ''

    while not (marker == 'X' or marker == '0'):
        marker = input('Player 1: Do you want to be X or O? ').upper()

        if marker == 'X':
            return ('X', '0')

        else:
            return ('0', 'X')
```

In [4]:

```
player_input()
```

```
Player 1: Do you want to be X or O? X
```

Out[4]:

```
('X', '0')
```

Step 3: Write a function that takes in the board list object, a marker ('X' or 'O'), and a desired position (number 1-9) and assigns it to the board.

In [5]:

```
def place_marker(board, marker, position):
    board[position] = marker
```

In [8]:

```
place_marker(test_board, '$', 8)
display_board(test_board)
```

```
   |   |
 X | $ | X
   |   |
-----------
   |   |
 0 | X | 0
   |   |
-----------
   |   |
 X | 0 | X
   |   |
```

In [ ]:

In [ ]:

Step 4: Write a function that takes in a board and checks to see if someone has won.

In [9]:

```
def win_check(board, mark):
    return ((board[7] == mark and board[8] == mark and board[9] == mark) or # accross the top(board[4] == mark an
d board[5] == mark and board[6] == mark) or # across the middle
    (board[4] == mark and board[5] ==mark and board[6] == mark ) or
    (board[1] == mark and board[2] == mark and board[3] == mark) or # across the bottom
    (board[7] == mark and board[4] == mark and board[1] == mark) or # down the middle
    (board[8] == mark and board[5] == mark and board[2] == mark) or #  down the middle
    (board[9] == mark and board[6] == mark and board[3] == mark) or # down the right side
    (board[7] == mark and board[5] == mark and board[3] == mark) or # diagonal
    (board[9] == mark and board[5] == mark and board[1] == mark))  # diagonal
```

In [10]:

```
win_check(test_board, 'X')
```

Out[10]:

```
True
```

Step 5: Write a function that uses the random module to randomly decide which player goes first. You may want to lookup random.randint() Return a string of which player went first.

In [11]:

```
import random

def choose_first():
    if random.randint(0, 1) == 0:
        return 'Player 2'

    else:
        return 'Player 1'
```

In [ ]:

Step 6: Write a function that returns a boolean indicating whether a space on the board is freely available.

In [12]:

```
def space_check(board, position):
    return board[position] == ' '
```

In [ ]:

In [ ]:

```
```

Step 7: Write a function that checks if the board is full and returns a boolean value. True if full, False otherwise

In [13]:

```
def full_board_check(board):
    for i in range(1,10):
        if space_check(board, i):
            return False
    return True
```

Step 8: Write a function that asks for a player's next position (as a number 1-9) ?

In [14]:

```
def player_choise(board):
    position = 0

    while position not in [1,2,3,4,5,6,7,8,9] or not space_check(board, position):
        position = int(input('Choose your next position : (1-9) '))

    return position
```

Step 9: Write a function that asks the player if they want to play again and returns a boolean True if they do want to play again.

In [15]:

```
def reply():
    return input('Do you want to play again ? Enter Yes or No: ').lower().startswith('y')
```

Step 10: Here comes the hard part! Use while loops and the functions you've made to run the game!

In [ ]:

```
print('Welcome to LetsUpgrade Tic Tac Toe Game')

while True:
    # Reset the board
    theBoard = [' '] * 10
    player1_marker, player2_marker = player_input()
    turn = choose_first()
    print(turn + ' will go first')

    play_game = input('Are you ready to play ? Enter Yes or No.')

    if play_game.lower()[0] == 'y':
        game_on = True
    else:
        game_on = False

    while game_on:
        if turn == 'Player 1':
            display_board(theBoard)
            position = player_choise(theBoard)
            place_marker(theBoard, player1_marker, position)

            if win_check(theBoard, player1_marker):
                display_board(theBoard)
                print('Congratulation! You have won the game')
                game_on = False

            else:
                if full_board_check(theBoard):
                    display_board(theBoard)
                    print('The game is a draw')
                    break
                else:
                    turn = 'Player 2'
        else:
            display_board(theBoard)
            position = player_choise(theBoard)
            place_marker(theBoard, player2_marker, position)

            if win_check(theBoard, player2_marker):
                display_board[theBoard]
                print('Player 2 has won')
                game_on = False

            else:
                if full_board_check(theBoard):
                    display_board(theBoard)
                    print('The game is a draw')
                    break
                else:
                    turn = 'Player 1'



    if not reply():
        break
```

```
Welcome to LetsUpgrade Tic Tac Toe Game
```

In [ ]:

```

```

In [ ]:

```

```