IronHack Week 1 Project

Build a game in Python

Boubacar Traoré & Clément Pilastre

Table of contents

01

Project Setup

Choosing the game - Planning the project - Development environment

Choosing the game

TIC
TAC
TOE

Planning the project

Using Jira

As the planning tool

Decomposing the steps

Between tasks
and subtasks

Assigning tasks

Between the
two of us

Development environment

Python

As the programing language

Visual Studio & Jupyter Notebook

As code editors

Github

For version control

02

The Algorithm

Decomposing the game - Flowchart

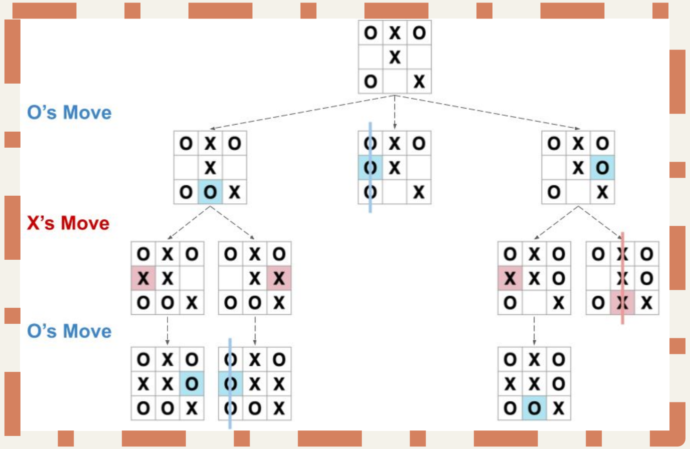
Tic Tac Toe is super simple right ?

NO, IT'S NOT.
SNL

Decomposing the game step by step

The obvious ones

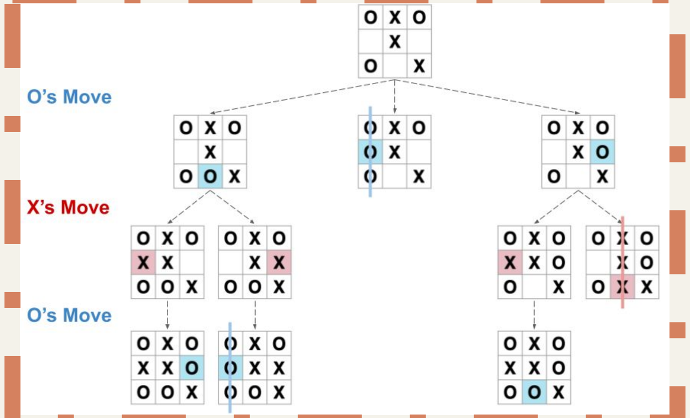
- Turn Based
- Placing X and O symbols
- Win when 3 symbols are aligned

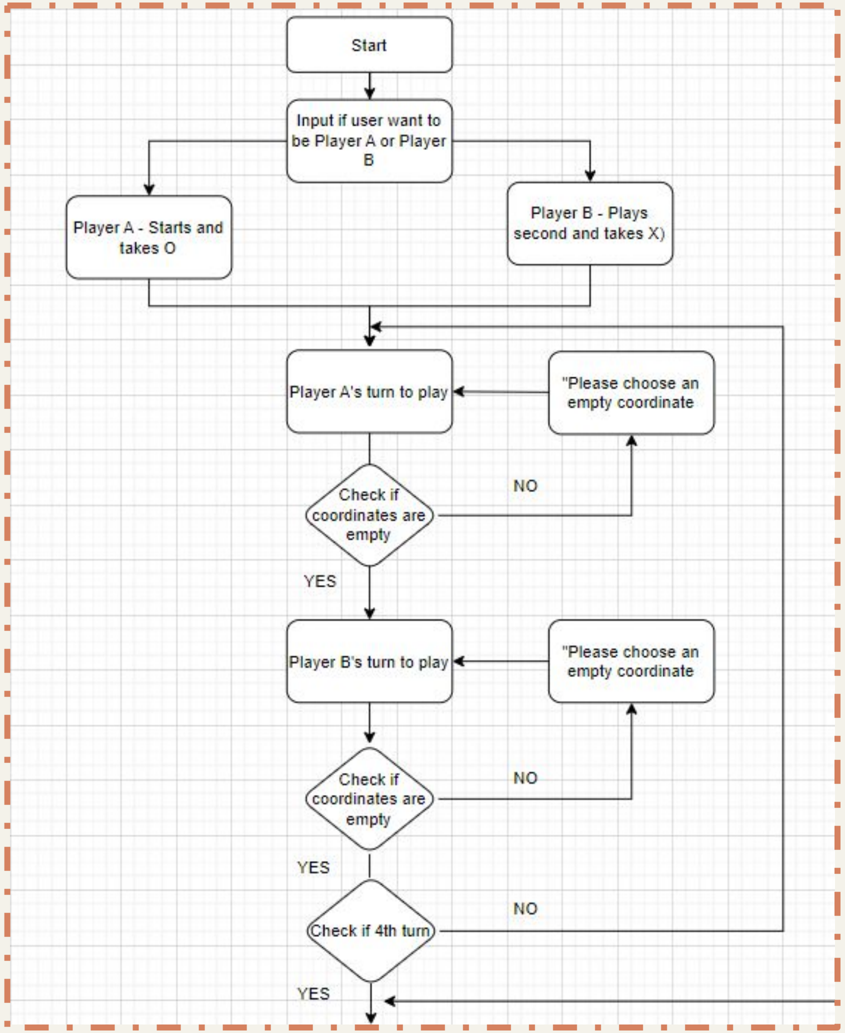


Decomposing the game step by step

The not-so obvious ones

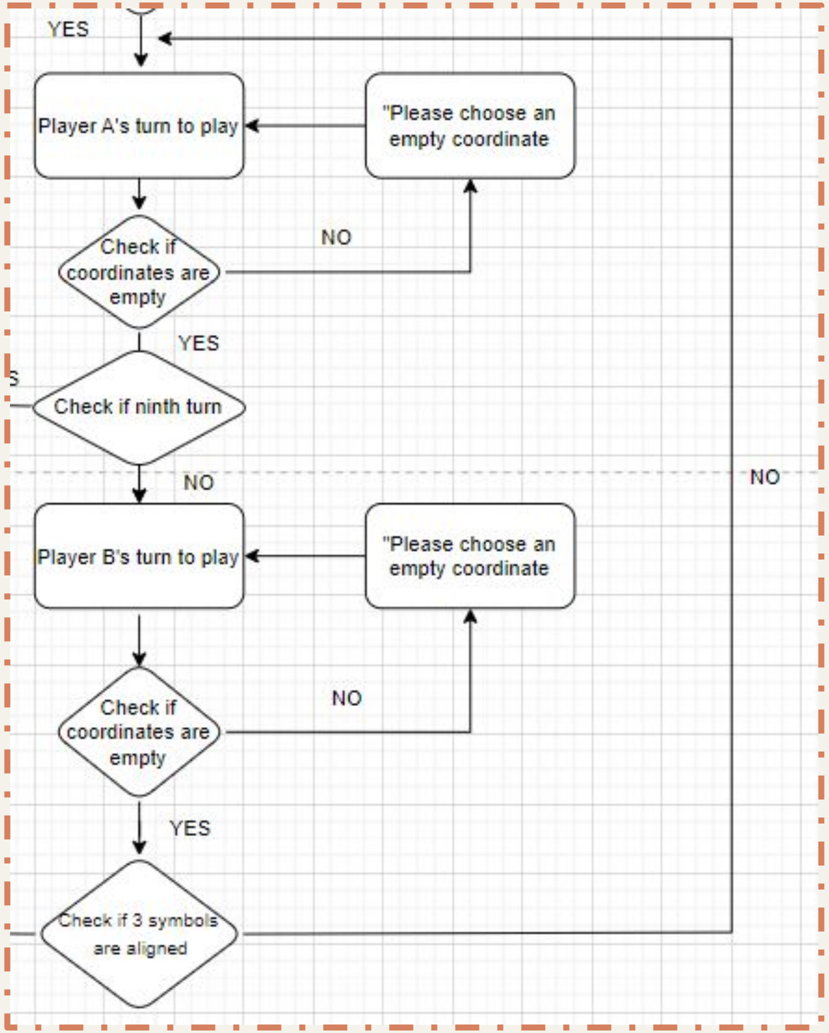
- Can’t place a symbols on an already occupied space
- Can’t place symbols outside the grid
- Automatic draw after 9 turns if 3 symbols aren’t aligned




Start
Input if user want to be Player A or Player B
Player A - Starts and takes O
Player B - Plays second and takes X)
Player A's turn to play
"Please choose an empty coordinate
Check if coordinates are empty
NO
YES
Player B's turn to play
"Please choose an empty coordinate
Check if coordinates are empty
NO
YES
Check if 4th turn
NO
YES

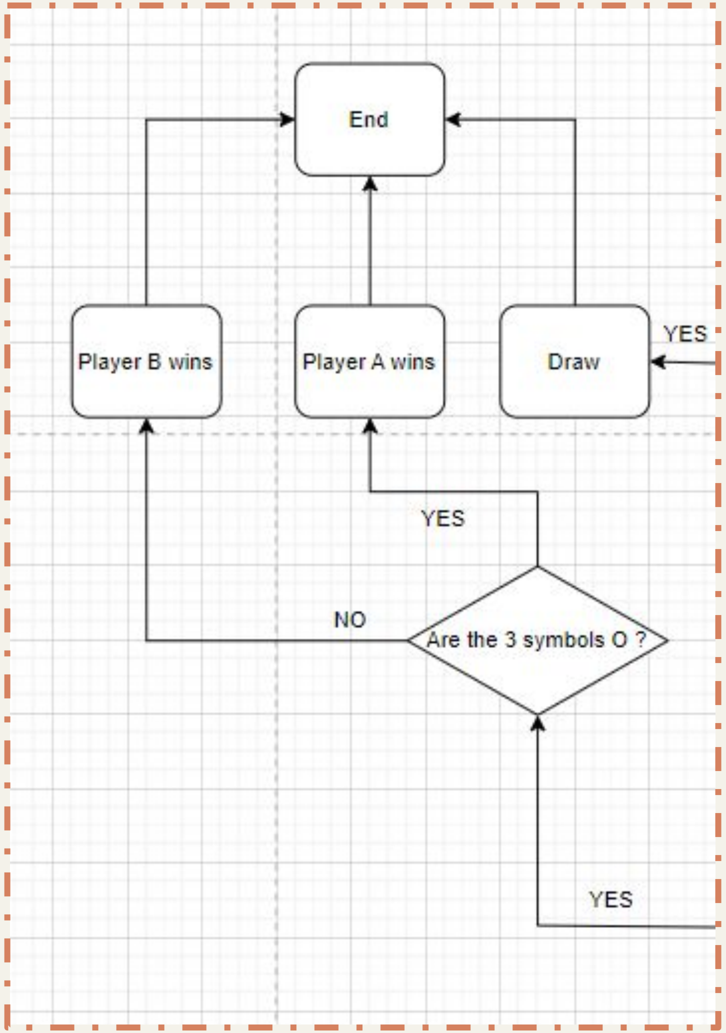
Flow chart First block

First block is straight forward, player choice and first turns with the exception of the important check to see if the space is empty


YES
Player A's turn to play
"Please choose an empty coordinate
Check if coordinates are empty
NO
YES
Check if ninth turn
NO
NO
Player B's turn to play
"Please choose an empty coordinate
Check if coordinates are empty
NO
YES
Check if 3 symbols are aligned

Flow chart Second block

We are still iterating through the turns while keeping the check for empty coordinates but on this block we implement a check for the win and the draw, starting respectively at the fourth and last turn



Flow chart Third block

After checking to see if the game is over now on the third block we check for the exact winner between the two players

03

Coding

Splitting the steps into functions

Splitting the steps into functions

Function “tictactoe”

Tasked with running the turn-based part

Function “visualization”

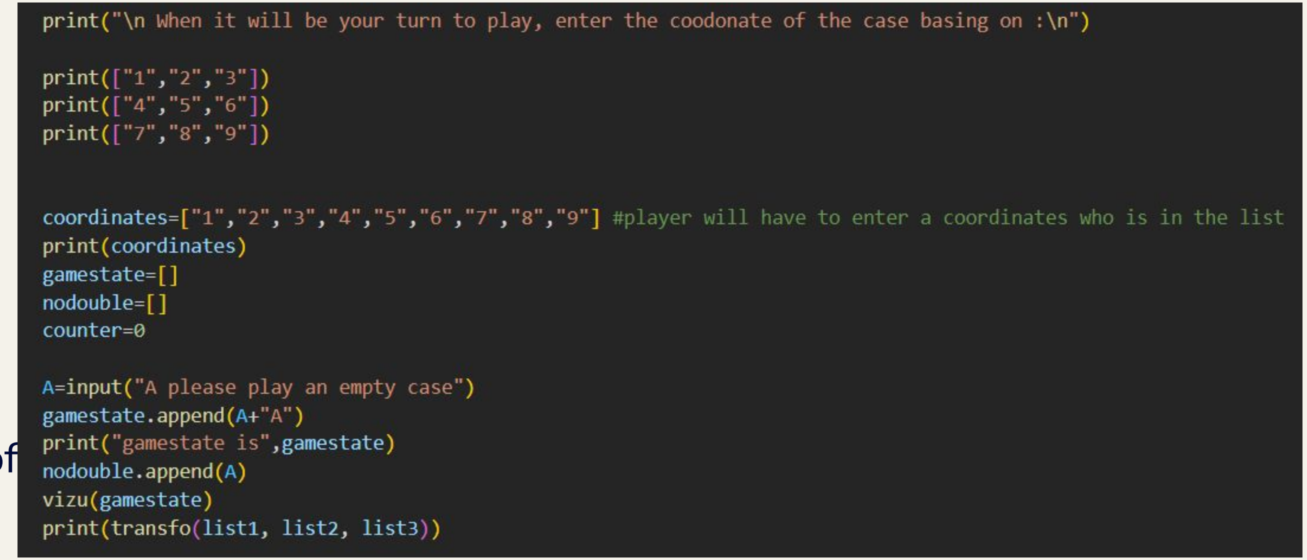
Tasked with representing the board after each turn

Function “whowon”

Tasked with returning the end result

Function Tictacoe I

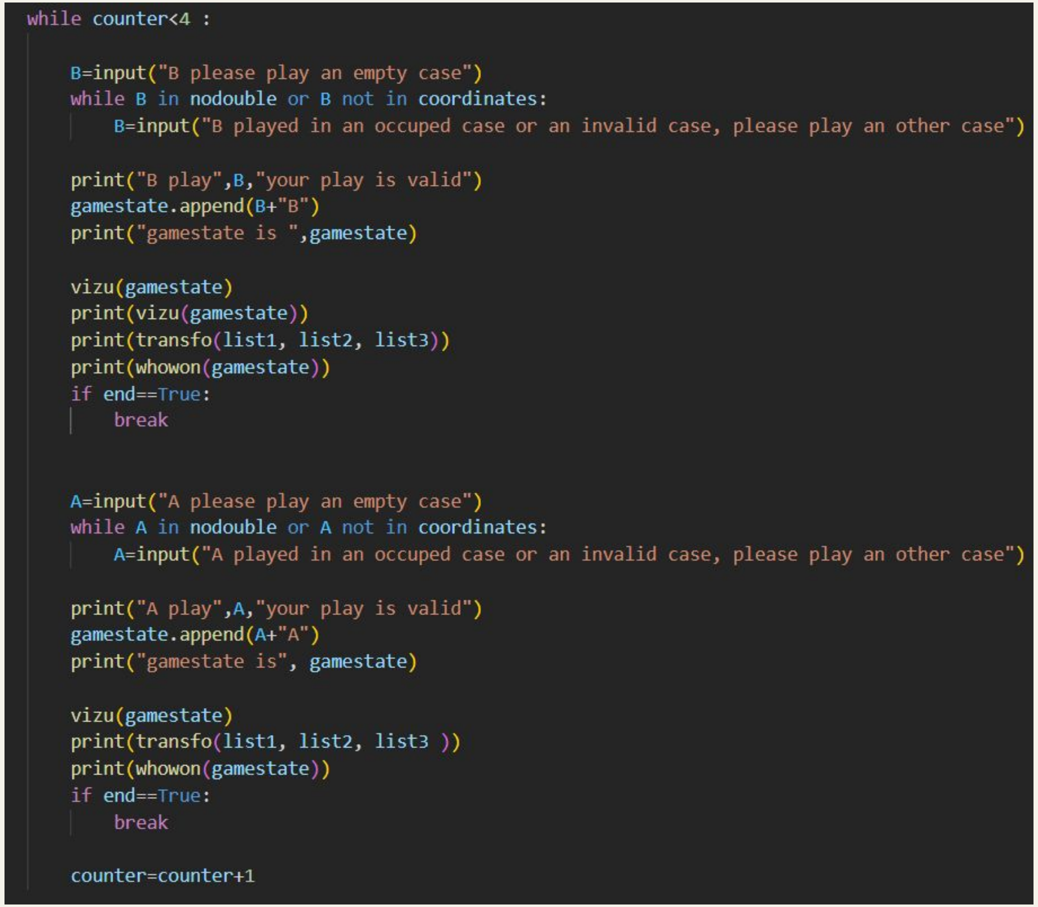
Tasked with iterating through the turns and asking the player to input the coordinates of his plays



```
print("\n When it will be your turn to play, enter the coodonate of the case basing on :\n")

print(["1","2","3"])
print(["4","5","6"])
print(["7","8","9"])


coordinates=["1","2","3","4","5","6","7","8","9"] #player will have to enter a coordinates who is in the list
print(coordinates)
gamestate=[]
nodouble=[]
counter=0

A=input("A please play an empty case")
gamestate.append(A+"A")
print("gamestate is",gamestate)
nodouble.append(A)
vizu(gamestate)
print(transfo(list1, list2, list3))
```

Function Tictacoe II

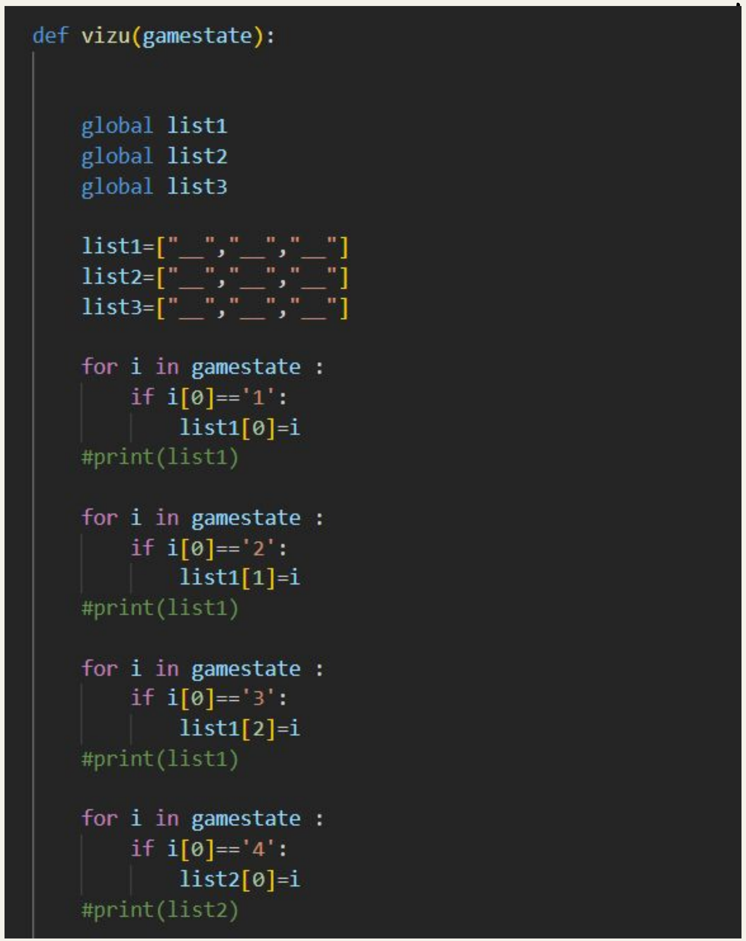
Tasked with iterating through the turns and asking the player to input the coordinates of his plays



```
while counter<4 :

    B=input("B please play an empty case")
    while B in nodouble or B not in coordinates:
        B=input("B played in an occuped case or an invalid case, please play an other case")

    print("B play",B,"your play is valid")
    gamestate.append(B+"B")
    print("gamestate is ",gamestate)

    vizu(gamestate)
    print(vizu(gamestate))
    print(transfo(list1, list2, list3))
    print(whowon(gamestate))
    if end==True:
        break


    A=input("A please play an empty case")
    while A in nodouble or A not in coordinates:
        A=input("A played in an occuped case or an invalid case, please play an other case")

    print("A play",A,"your play is valid")
    gamestate.append(A+"A")
    print("gamestate is", gamestate)

    vizu(gamestate)
    print(transfo(list1, list2, list3 ))
    print(whowon(gamestate))
    if end==True:
        break

    counter=counter+1
```

Functions Visualization

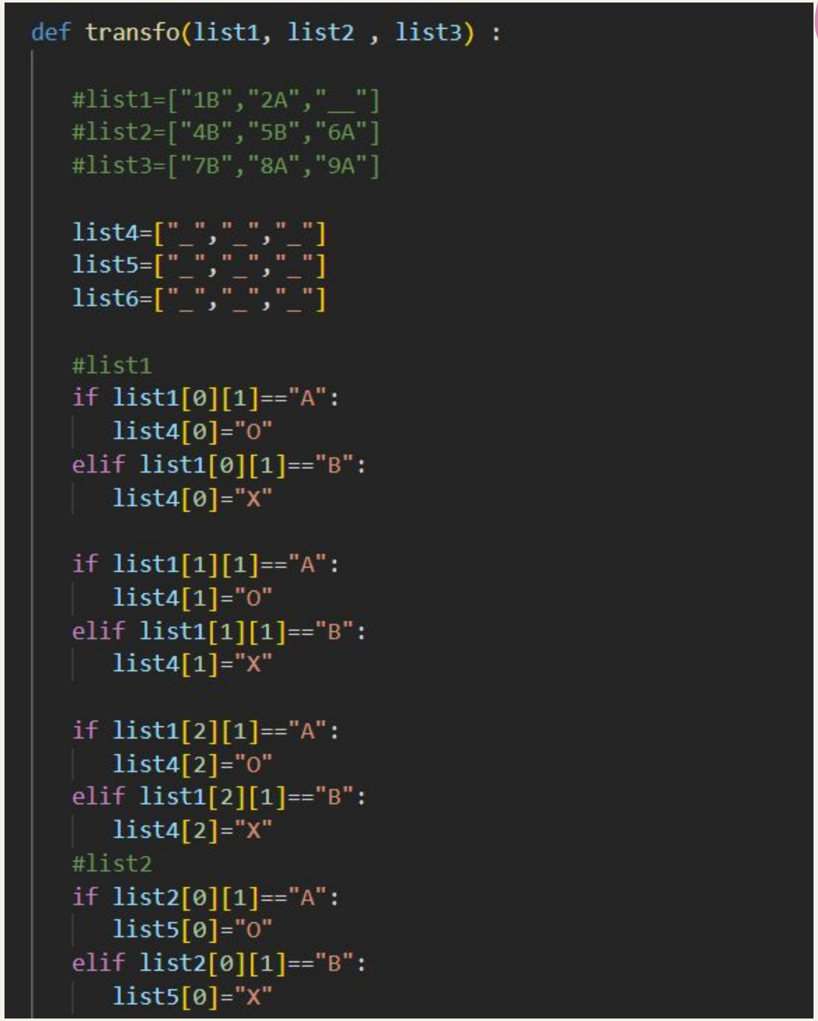
Tasked with representing the grid after each turn



```python
def vizu(gamestate):


    global list1
    global list2
    global list3

    list1=["__","__","__"]
    list2=["__","__","__"]
    list3=["__","__","__"]

    for i in gamestate :
        if i[0]=='1':
            list1[0]=i
    #print(list1)

    for i in gamestate :
        if i[0]=='2':
            list1[1]=i
    #print(list1)

    for i in gamestate :
        if i[0]=='3':
            list1[2]=i
    #print(list1)

    for i in gamestate :
        if i[0]=='4':
            list2[0]=i
    #print(list2)
```

Functions Transformation

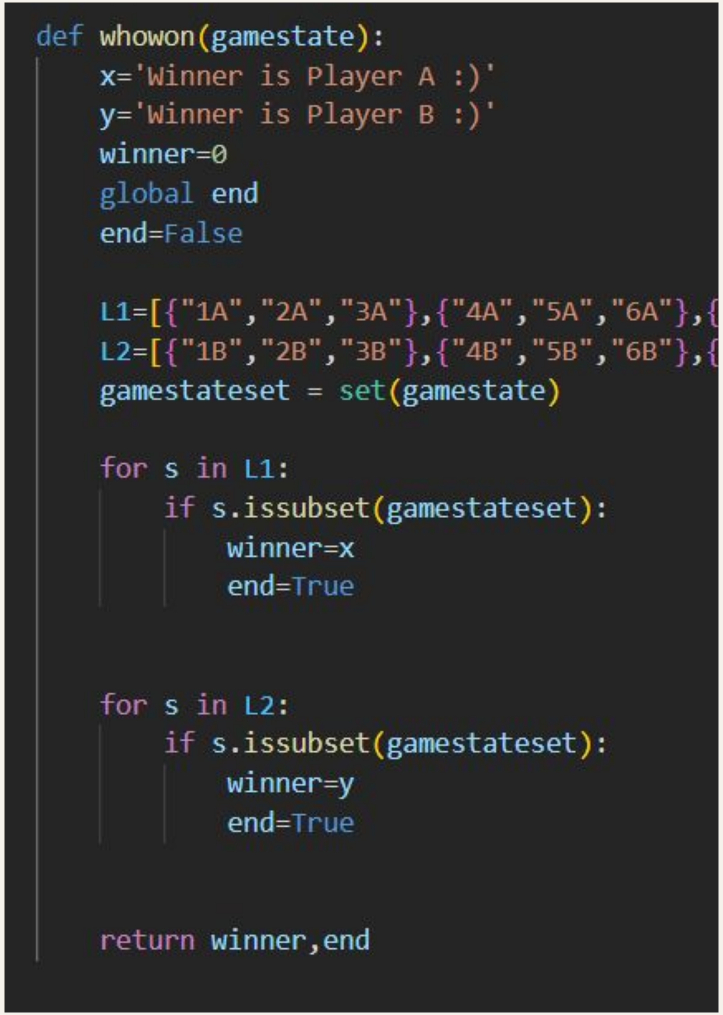
Tasked with representing the X and O in the grid after each turn



```
def transfo(list1, list2 , list3) :

    #list1=["1B","2A","__"]
    #list2=["4B","5B","6A"]
    #list3=["7B","8A","9A"]

    list4=["_","_","_"]
    list5=["_","_","_"]
    list6=["_","_","_"]

    #list1
    if list1[0][1]=="A":
        list4[0]="O"
    elif list1[0][1]=="B":
        list4[0]="X"

    if list1[1][1]=="A":
        list4[1]="O"
    elif list1[1][1]=="B":
        list4[1]="X"

    if list1[2][1]=="A":
        list4[2]="O"
    elif list1[2][1]=="B":
        list4[2]="X"
    #list2
    if list2[0][1]=="A":
        list5[0]="O"
    elif list2[0][1]=="B":
        list5[0]="X"
```

Function Whowon

Tasked with returning the winner starting



```
def whowon(gamestate):
    x='Winner is Player A :)'
    y='Winner is Player B :)'
    winner=0
    global end
    end=False

    L1=[{"1A","2A","3A"},{"4A","5A","6A"},{
    L2=[{"1B","2B","3B"},{"4B","5B","6B"},{
    gamestateset = set(gamestate)

    for s in L1:
        if s.issubset(gamestateset):
            winner=x
            end=True


    for s in L2:
        if s.issubset(gamestateset):
            winner=y
            end=True


    return winner,end
```

Challenges

Assembling the code

Making all our functions works together was more challenging than we thought

Juggling between data structures types

Juggling between the data structures to use the advantages of all of them

Display a tictactoe grid

Find a roundabout way to display a grid from 3 lists

04

Game Demonstration

05

Q and A

Please be kind :-)