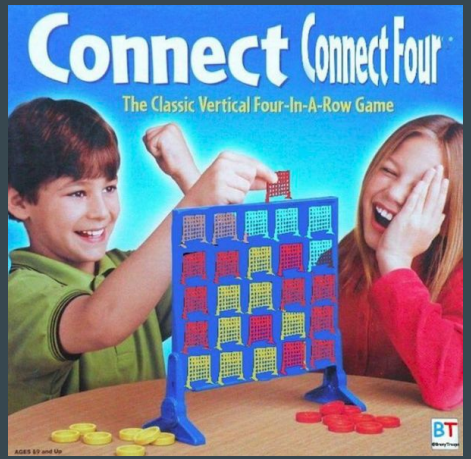
AP1 PROJECT 2018-19
CONNECT 4

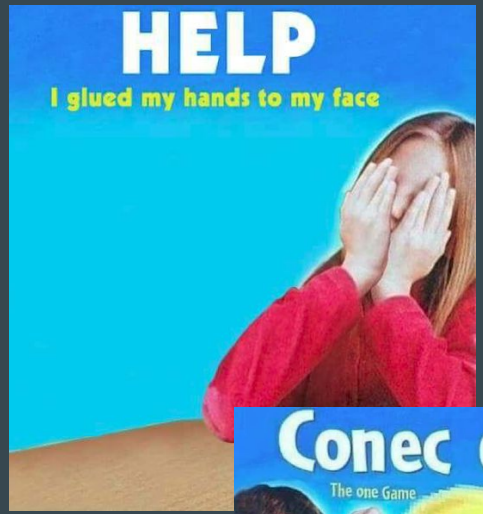
...

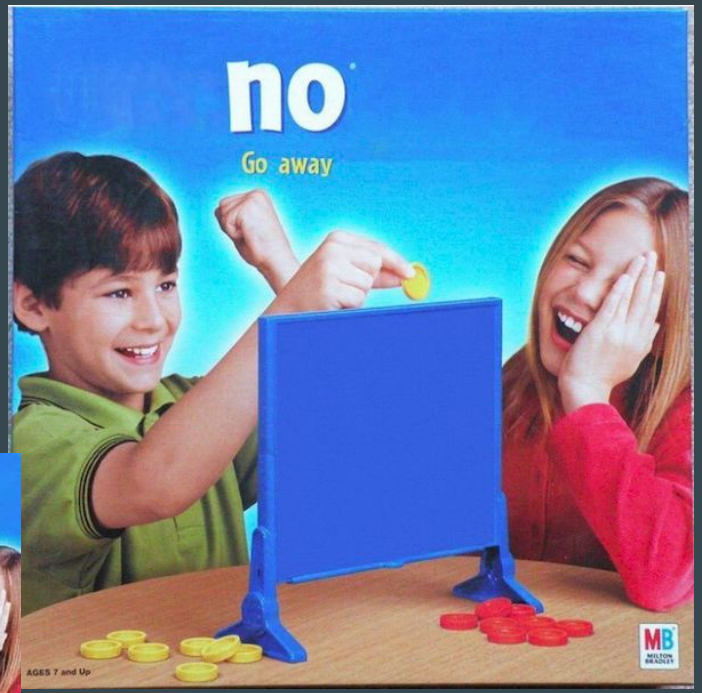
Bilal Belfodil | Antoine Nollet


Connect Four
The Classic Vertical Four-In-A-Row Game
AGES 7 and Up
MB
MILTON BRADLEY

Some CONNECT 4 memes ..



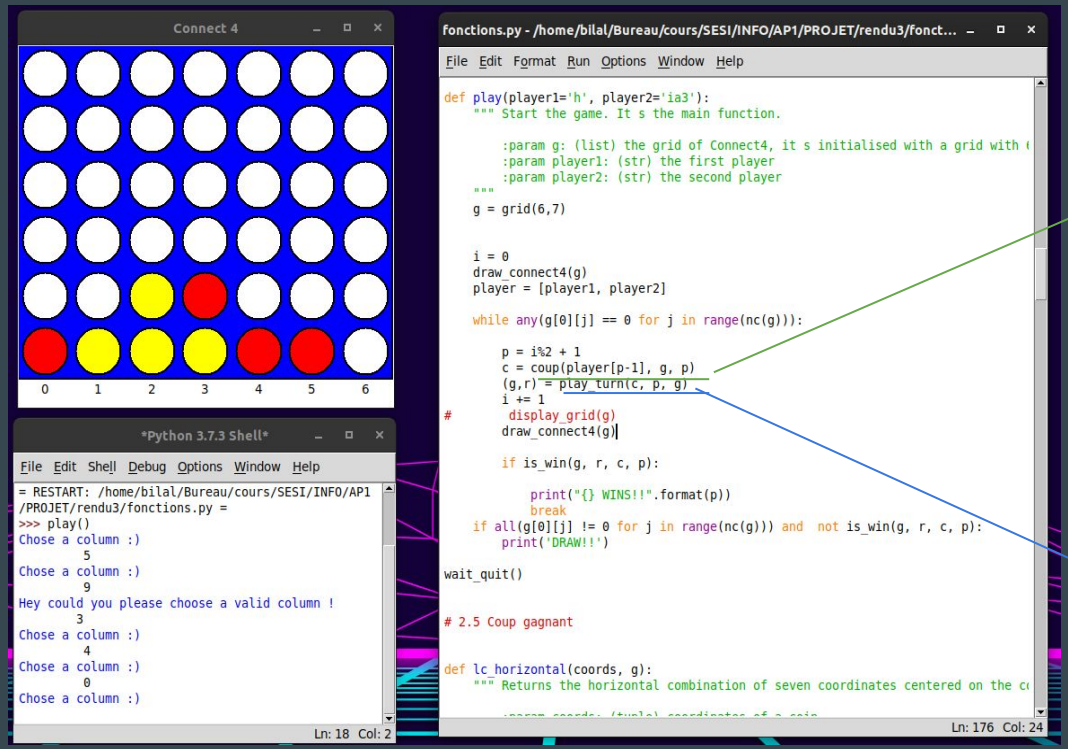




Okay, but WHY did we choose CONNECT 4 ?

- A funny game
- Easily understandable principle
- Simple rules
- Well known
- Simple rules implies complex strategies

How it works

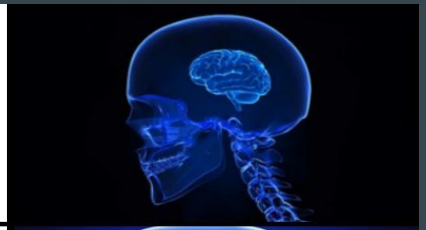



```python
def coup(player, g, p):
    """  Allows the player to play

        :param player: (str) the player
        :param g: (list) the grid of Connect4
        :param p: (int) A or 2 depending on the player
        :return: (function) the playing function
    """

    if player == 'h':

        return ask_player(g)

    elif player == 'ia1':

        return ia_aleat(g)

    elif player == 'ia2':

        return ia_win(g, p)

    elif player == 'ia3':

        return ia_win2(g,p)

    elif player == 'ia4':

        return move_ia(g ,3, p)
```

```python
def play_turn(c, p, g):
    """ Modifies the grid at each round

        :param c: (int) the chosen column
        :param p: (int) the player
        :param g: (list) the grid
        :return: (tuple) the row and the modified grid
    """

    r= 0

    for i in range(nr(g)):
        if g[i][c] != 0:
            r += 1

    g[nr(g)-r-1][c] = p

    return (g, nr(g)-r-1)
```

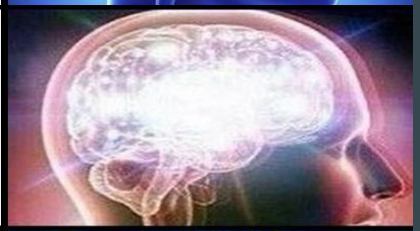
THE AIs LEVELS

Child : plays randomly



Smart child : plays randomly except if it can win

Professor : A smarter one which chooses the best column for this round (sometimes cheats)



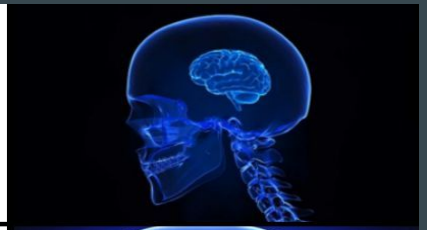
Einstein : Analyses the best choice for some rounds in the future

```
def ia_aleat(g):
    """ Makes the ia plays randomly.

        :param g: (list) the grid of Connect4
        :return: (int) the column where the ia is going to play, chosen randomly
    """

    liste_coups = [i for i in range(nc(g)) if is_valid(i, g)]

    return liste_coups[randrange(len(liste_coups))]
```

```
def ia_win(g, p):
    """ Makes the ia checks if his playturn could be winning. If there a poss
        Else, return a random column.

        :param g: (list) the grid of Connect4
        :param p: (int) the value corresponding of the coin s player
        :return: (int) the number of a column, corresponding to a winning one
    """


    copy = list.copy(g)

    for i in range(nc(copy)):

        (copy, r) = play_turn(i, p, copy)

        if is_win(g, r, i, p):

            unmove(copy, i)
            return i

        else:

            unmove(copy, i)

    return ia_aleat(g)
```

THE AIs LEVELS

Child : plays randomly



Smart child : plays randomly except if it can win

Professor : A smarter one which chooses the best column for this round (sometimes cheats)

Einstein : Analyses the best choice for some rounds in the future

```
def OptimusColumn(g,p):
    """ Returns the column which give us the best play.

        :param g: (list) the grid of Connect4
        :param p: (int) the player
        :return: (int) the best column
    """
    listScore = []
    copy = list.copy(g)
    for c in range(nc(copy)):
        if is_valid(c, copy):
            (copy, r) = play_turn(c,p, copy)
            listScore.append(computeGrid(copy, p))
            unmove(copy, c)
        else:
            listScore.append(-1000000000000000000000000000000000000000000000000)

    return listScore.index(max(listScore))


def ia_win2(g, p):
    """ Makes the ia checks if his playturn could be winning. If there a possibil
        Else, return a random column.

        :param g: (list) the grid of Connect4
        :param p: (int) the value corresponding of the coin s player
        :return: (int) the number of a column, corresponding to the best playturn
    """
    copy = list.copy(g)
    for i in range(nc(copy)):
        (copy, r) = play_turn(i, p, copy)

        if is_win(g, r, i, p):
            unmove(copy, i)
            return i
        else:
            unmove(copy, i)
    return OptimusColumn(g,p)
```

Interface

```
from tkinter import *

class Main_Menu(Frame):

    """Notre fenêtre principale.
    Tous les widgets sont stockés comme attributs de cette fenêtre."""

    def __init__(self, fenetre, **kwargs):
        Frame.__init__(self, fenetre, width=768, height=576, **kwargs)
        self.pack(fill=BOTH)
        self.nb_clic = 0

        # Création de nos widgets
        self.message = Label(self, text="CONNECT 4 !!")
        self.message.pack()

        self.play_button = Button(self, text="Play", command=self.Play)
        self.play_button.pack()

        self.quit_button = Button(self, text="Quit", command=self.quit)
        self.quit_button.pack()



    def  Play(self):
        interface = Playing_menu(fenetre)


class Playing_menu(Frame):

    """"The playing menu where the player chooses how he wants to play"""

    def __init__(self, fenetre, **kwargs):
        Frame.__init__(self, fenetre, width=768, height=576, **kwargs)
        self.pack(fill=BOTH)

        self.message = Label(self, text="CONNECT 4 !!")
        self.message.pack()

        self.bouton_quitter = Button(self, text="Back", command=self.quit)
        self.bouton_quitter.pack()




fenetre = Tk()
interface = Main_Menu(fenetre)

interface.mainloop()
interface.destroy()
```

```
def display():
    for elt in screen:
        print(elt,end = '')

def change_ecran(i):


    L = LL[i]
    return ["            \  /\n",
"              \/\n",
"    .====================.\n",
"    | .----------------. |\n",
"    | |                | |\n",
"    | |","{:^16}".format(L[0]),"| |\n",
"    | |","{:^16}".format(L[1]),"| |\n",
"    | |","{:^16}".format(L[2]),"| |\n",
"    | |","{:^16}".format(L[3]),"| |\n",
"    | '----------------'o|\n",
"    |====================|\n",
"    |####################|\n",
"    '===================='\n"]


display()
input()

i = 1
screen = change_ecran(i)
display()

choix = int(input())

if choix == 2:
    i = 2
screen = change_ecran(i)
display()
```

Let’s try the game !!