Lab 2 AI

Connect 4

Presented to

Dr Marwan el torki

Ta eng.mohamed el habibi

OMAR ELSAMMAK

Overview:

Connect 4 is a two-player game in which the players first choose a color and then take turns dropping their colored discs from the top into a grid. The pieces fall straight down, occupying the next available space within the column. The objective of the game is to connect-four of one's own discs of the same color next to each other vertically, horizontally, or diagonally. The two players keep playing until the board is full. The winner is the player having greater number of connected-fours.

Functions used:

1. Draw_gui

   This function draws the array to form a gui to make communication easier
2. Array_2_int

   This function takes the array that represents the board and stores it in a Long int to store the game state (it represents each column state in 9 bits)
3. Int_2_array

   This function takes the stored long int and transfers it into a 6*7 array to represent the state of the board
4. Get_playable_row

   This function takes the column we want to play in and the state of the board as a long int and gets which row is the last free one so that we can drop the piece in it
5. Get_playable_columns

   This function takes the board state as a long int and checks which columns are available
6. Drop_checker

   This functions take the row and column we want to play in and which turn is it and it places the checker piece in the desired place
7. Print_arr

   Takes the state as and array and prints it just to illustrate and make sure that the gui is

   Working (just a check not important)
8. Calculate_score

   Calculates the weight of the given node so that we can use it in the minimax algorithm
9. Get_fours

   This functions traverses through the array to get how many combinations of fours are there for a certain player (either player 1 or 2) to be used when calculating the score of a certain node , also used to check which player is the winner

10. Get_threes

    This functions traverses through the array to get how many combinations of threes are there for a certain player (either player 1 or 2) to be used when calculating the score of a certain node

11. Get_twos

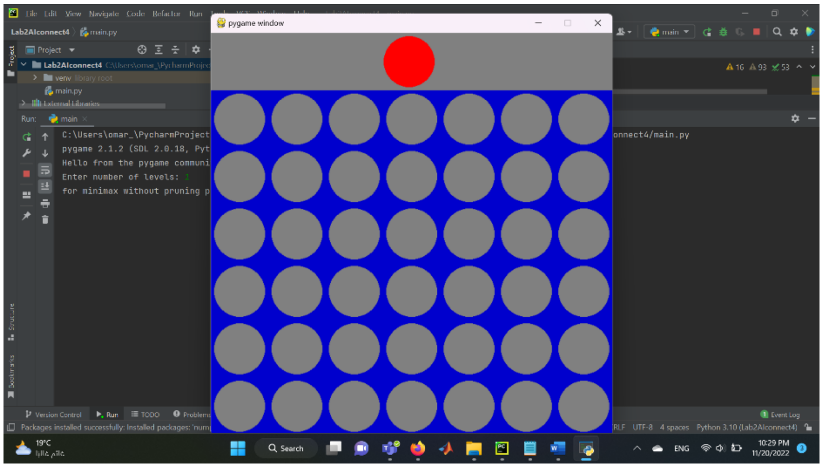
    This functions traverses through the array to get how many combinations of twos are there for a certain player (either player 1 or 2) to be used when calculating the score of a certain node

12. Minimax

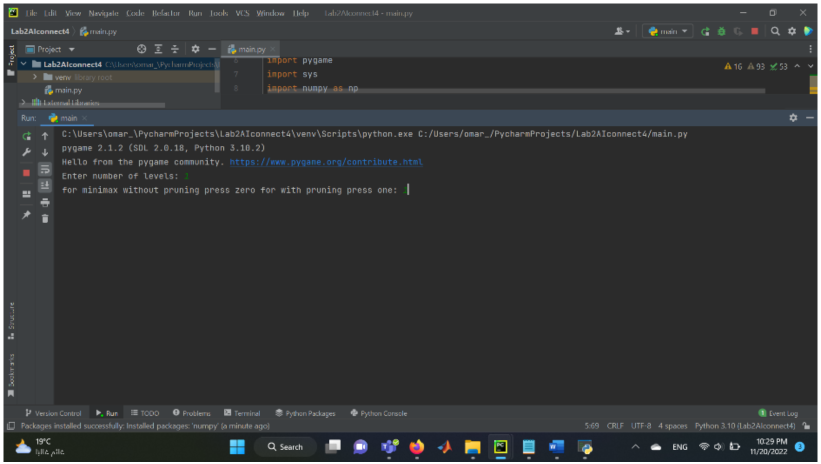
    the main algorithm used for making the computer play against a user
    it is a recursive algorithm ,it also uses recursion to search through the game-tree.
    two players play the game, one is called MAX and other is called MIN
    Both Players of the game are opponent of each other, where MAX will select the maximized value and MIN will select the minimized value
    This algorithm applies DFS, so in this game-tree, we have to go all the way through the leaves to reach the terminal nodes
    This function takes the depth of the tree from the user as an input K
    And whether we want to use it using alpha-beta pruning or not

13. If_game_ened

    This function checks if all the columns are full to see if the game ended or not

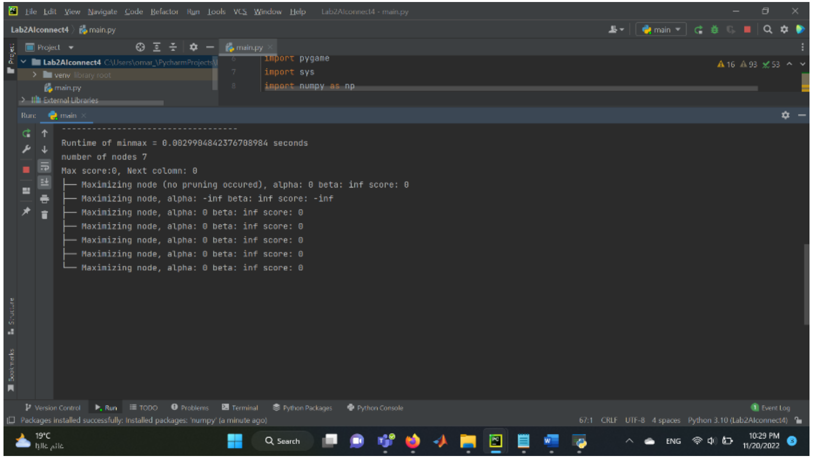


```
C:\Users\omar_\PycharmProject                                              onnect4/main.py
pygame 2.1.2 (SDL 2.0.18, Pyt
Hello from the pygame communi
Enter number of levels: 1
for minimax without pruning p
```



```
import pygame
import sys
import numpy as np
```

```
C:\Users\omar_\PycharmProjects\Lab2AIconnect4\venv\Scripts\python.exe C:/Users/omar_/PycharmProjects/Lab2AIconnect4/main.py
pygame 2.1.2 (SDL 2.0.18, Python 3.10.2)
Hello from the pygame community. https://www.pygame.org/contribute.html
Enter number of levels: 1
for minimax without pruning press zero for with pruning press one: 1
```



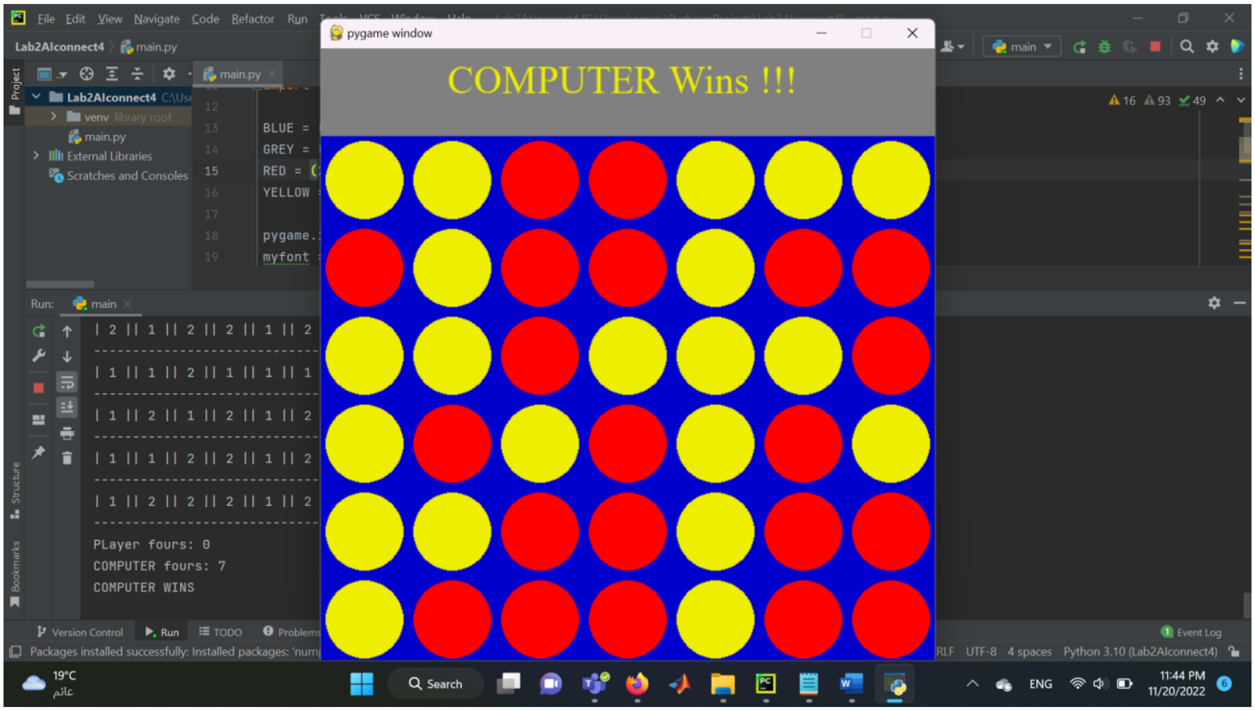
```
import pygame
import sys
import numpy as np
```

```
-----------------------------------
Runtime of minmax = 0.0029904842376708984 seconds
number of nodes 7
Max score:0, Next colomn: 0
 ├── Maximizing node (no pruning occured), alpha: 0 beta: inf score: 0
 ├── Maximizing node, alpha: -inf beta: inf score: -inf
 ├── Maximizing node, alpha: 0 beta: inf score: 0
 ├── Maximizing node, alpha: 0 beta: inf score: 0
 ├── Maximizing node, alpha: 0 beta: inf score: 0
 ├── Maximizing node, alpha: 0 beta: inf score: 0
 └── Maximizing node, alpha: 0 beta: inf score: 0
```

This is a sample run

The computer asks for the depth and whether we want to play using alpha-beta pruning or without, Then we can start playing using the gui

After every move the number of nodes expanded are printed,and the search tree is also printed



COMPUTER Wins !!!

```
BLUE =
GREY =
RED = (
YELLOW =

pygame.
myfont =
```

```
| 2 || 1 || 2 || 2 || 1 || 2
------------------------------
| 1 || 1 || 2 || 1 || 1 || 1
------------------------------
| 1 || 2 || 1 || 2 || 1 || 2
------------------------------
| 1 || 1 || 2 || 2 || 1 || 2
------------------------------
| 1 || 2 || 2 || 2 || 1 || 2
------------------------------
PLayer fours: 0
COMPUTER fours: 7
COMPUTER WINS
```

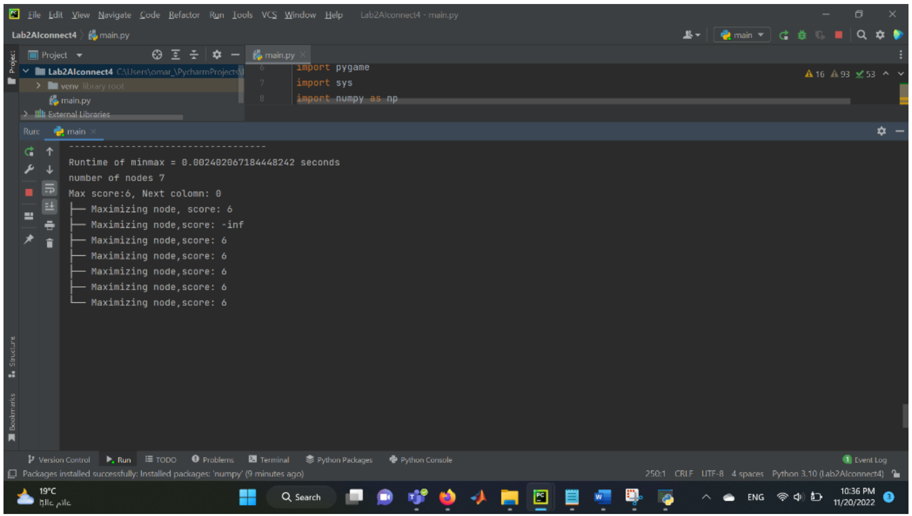
Packages installed successfully: Installed packages: 'num

Here is a sample run using depth one and with alpha beta pruning

Now I will demonstrate more by showing multiple test cases for multiple depths with and without alpha beta pruning

Without alpha beta pruning :

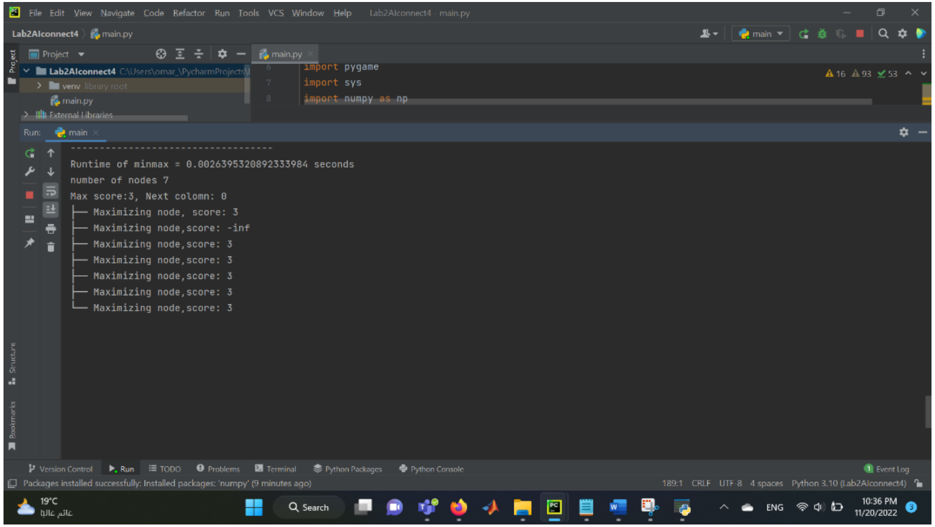
Depth 1



```
import pygame
import sys
import numpy as np
```

```
-----------------------------------
Runtime of minmax = 0.002402067184448242 seconds
number of nodes 7
Max score:6, Next colomn: 0
├── Maximizing node, score: 6
├── Maximizing node,score: -inf
├── Maximizing node,score: 6
├── Maximizing node,score: 6
├── Maximizing node,score: 6
├── Maximizing node,score: 6
└── Maximizing node,score: 6
```

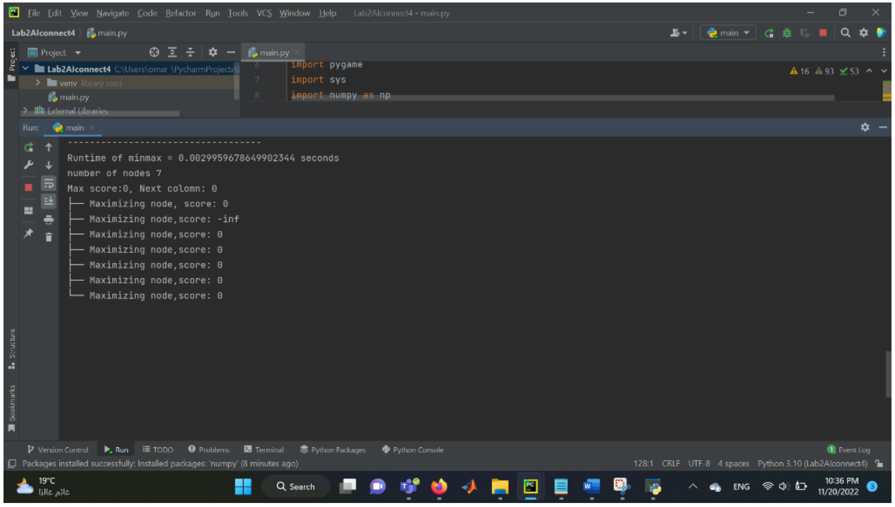
Packages installed successfully: Installed packages: 'numpy' (9 minutes ago)



```
import pygame
import sys
import numpy as np
```

```
-----------------------------------
Runtime of minmax = 0.0026395320892333984 seconds
number of nodes 7
Max score:3, Next colomn: 0
├── Maximizing node, score: 3
├── Maximizing node,score: -inf
├── Maximizing node,score: 3
├── Maximizing node,score: 3
├── Maximizing node,score: 3
├── Maximizing node,score: 3
└── Maximizing node,score: 3
```

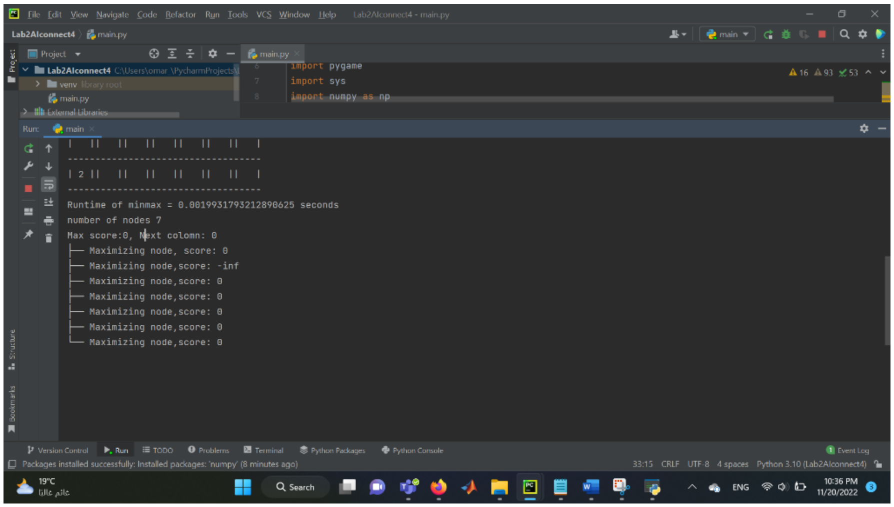
Packages installed successfully: Installed packages: 'numpy' (9 minutes ago)



```
import pygame
import sys
import numpy as np
```

```
-----------------------------------
Runtime of minmax = 0.0029959678649902344 seconds
number of nodes 7
Max score:0, Next colomn: 0
├── Maximizing node, score: 0
├── Maximizing node,score: -inf
├── Maximizing node,score: 0
├── Maximizing node,score: 0
├── Maximizing node,score: 0
├── Maximizing node,score: 0
└── Maximizing node,score: 0
```

Packages installed successfully: Installed packages: 'numpy' (8 minutes ago)



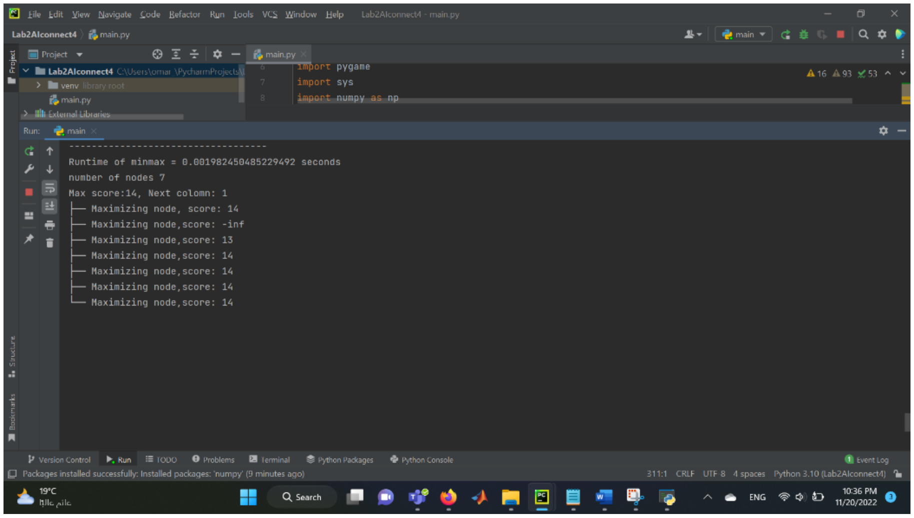
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Lab2AIconnect4 - main.py
Lab2AIconnect4 main.py
Project
Lab2AIconnect4 C:\Users\omar_\PycharmProjects\
venv library root
main.py
External Libraries
Run: main
Version Control Run TODO Problems Terminal Python Packages Python Console Event Log

```
import pygame
import sys
import numpy as np
```

```
|   ||   ||   ||   ||   ||   ||   |
-----------------------------------
| 2 ||   ||   ||   ||   ||   ||   |
-----------------------------------
Runtime of minmax = 0.0019931793212890625 seconds
number of nodes 7
Max score:0, Next colomn: 0
├── Maximizing node, score: 0
├── Maximizing node,score: -inf
├── Maximizing node,score: 0
├── Maximizing node,score: 0
├── Maximizing node,score: 0
├── Maximizing node,score: 0
└── Maximizing node,score: 0
```

Packages installed successfully: Installed packages: 'numpy' (8 minutes ago)

33:15 CRLF UTF-8 4 spaces Python 3.10 (Lab2AIconnect4)



File Edit View Navigate Code Refactor Run Tools VCS Window Help
Lab2AIconnect4 - main.py
Lab2AIconnect4 main.py
Project
Lab2AIconnect4 C:\Users\omar_\PycharmProjects\
venv library root
main.py
External Libraries
Run: main
Version Control Run TODO Problems Terminal Python Packages Python Console Event Log

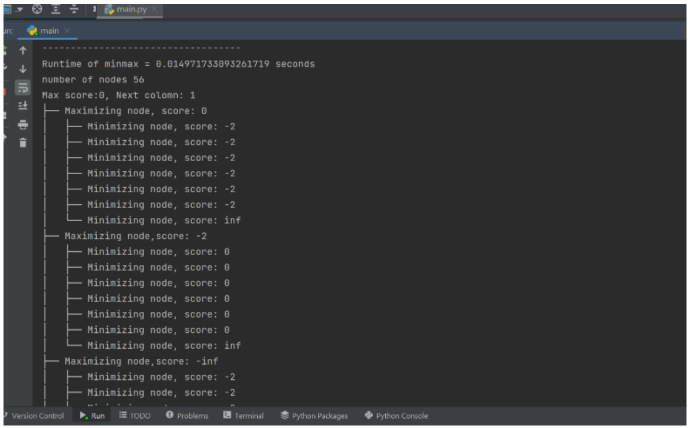
```
import pygame
import sys
import numpy as np
```

```
-----------------------------------
Runtime of minmax = 0.001982450485229492 seconds
number of nodes 7
Max score:14, Next colomn: 1
├── Maximizing node, score: 14
├── Maximizing node,score: -inf
├── Maximizing node,score: 13
├── Maximizing node,score: 14
├── Maximizing node,score: 14
├── Maximizing node,score: 14
└── Maximizing node,score: 14
```

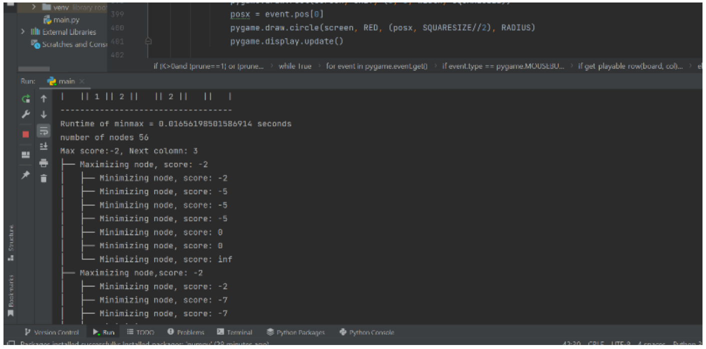
Packages installed successfully: Installed packages: 'numpy' (9 minutes ago)

311:1 CRLF UTF-8 4 spaces Python 3.10 (Lab2AIconnect4)

Depth 2:

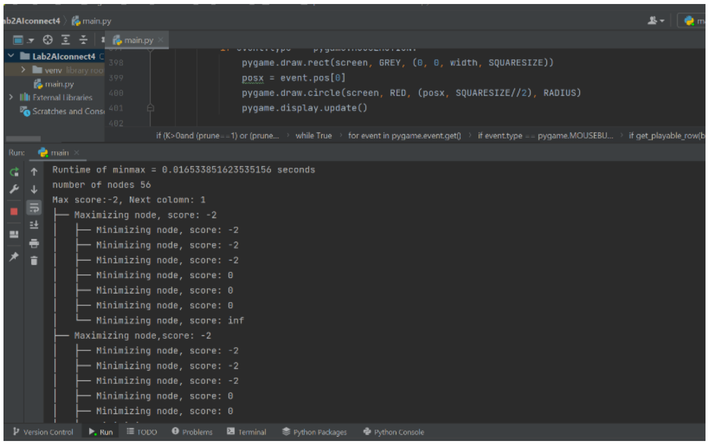


```
----------------------------------
Runtime of minmax = 0.014971733093261719 seconds
number of nodes 56
Max score:0, Next colomn: 1
├── Maximizing node, score: 0
│   ├── Minimizing node, score: -2
│   ├── Minimizing node, score: -2
│   ├── Minimizing node, score: -2
│   ├── Minimizing node, score: -2
│   ├── Minimizing node, score: -2
│   ├── Minimizing node, score: -2
│   └── Minimizing node, score: inf
├── Maximizing node,score: -2
│   ├── Minimizing node, score: 0
│   ├── Minimizing node, score: 0
│   ├── Minimizing node, score: 0
│   ├── Minimizing node, score: 0
│   ├── Minimizing node, score: 0
│   ├── Minimizing node, score: 0
│   └── Minimizing node, score: inf
├── Maximizing node,score: -inf
│   ├── Minimizing node, score: -2
│   ├── Minimizing node, score: -2
```



```
                posx = event.pos[0]
                pygame.draw.circle(screen, RED, (posx, SQUARESIZE//2), RADIUS)
                pygame.display.update()
```

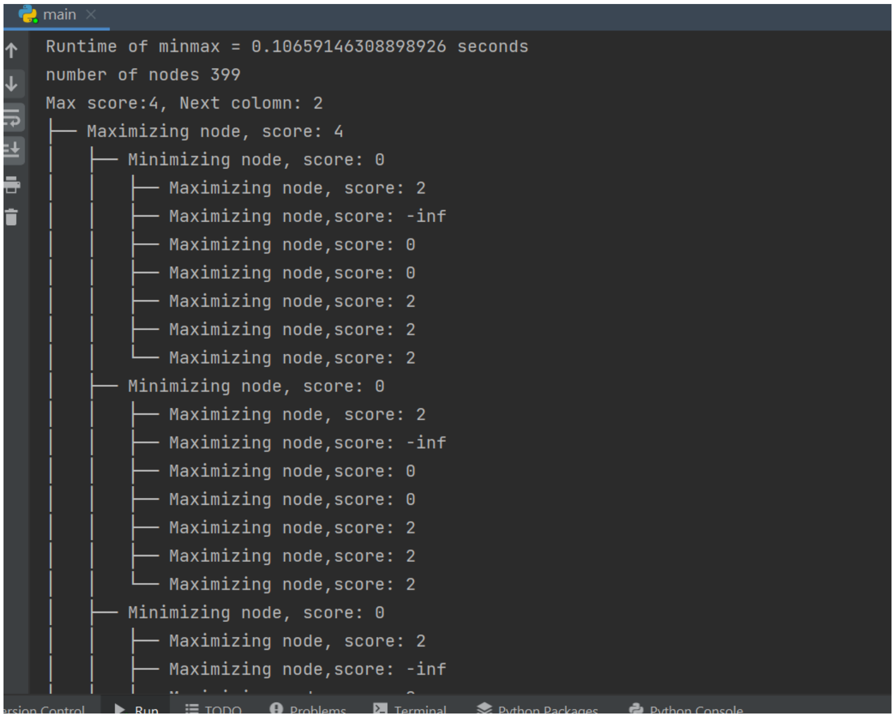
```
|   || 1 || 2 ||   || 2 ||   ||   |
----------------------------------
Runtime of minmax = 0.01656198501586914 seconds
number of nodes 56
Max score:-2, Next colomn: 3
├── Maximizing node, score: -2
│   ├── Minimizing node, score: -2
│   ├── Minimizing node, score: -5
│   ├── Minimizing node, score: -5
│   ├── Minimizing node, score: -5
│   ├── Minimizing node, score: 0
│   ├── Minimizing node, score: 0
│   └── Minimizing node, score: inf
├── Maximizing node,score: -2
│   ├── Minimizing node, score: -2
│   ├── Minimizing node, score: -7
│   ├── Minimizing node, score: -7
```



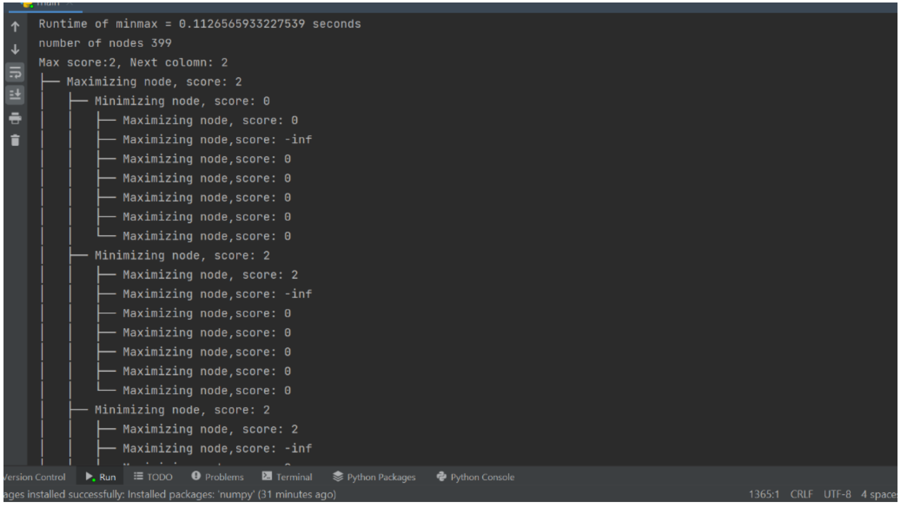
```
                pygame.draw.rect(screen, GREY, (0, 0, width, SQUARESIZE))
                posx = event.pos[0]
                pygame.draw.circle(screen, RED, (posx, SQUARESIZE//2), RADIUS)
                pygame.display.update()
```

```
Runtime of minmax = 0.016533851623535156 seconds
number of nodes 56
Max score:-2, Next colomn: 1
├── Maximizing node, score: -2
│   ├── Minimizing node, score: -2
│   ├── Minimizing node, score: -2
│   ├── Minimizing node, score: -2
│   ├── Minimizing node, score: 0
│   ├── Minimizing node, score: 0
│   ├── Minimizing node, score: 0
│   └── Minimizing node, score: inf
├── Maximizing node,score: -2
│   ├── Minimizing node, score: -2
│   ├── Minimizing node, score: -2
│   ├── Minimizing node, score: -2
│   ├── Minimizing node, score: 0
│   ├── Minimizing node, score: 0
```

Depth 3:



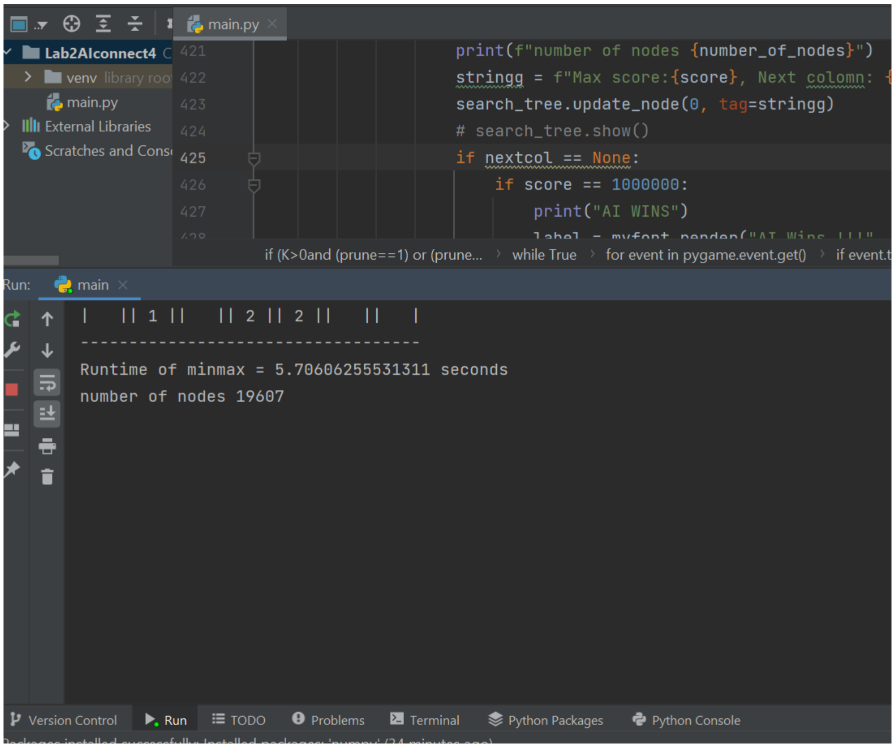
```
Runtime of minmax = 0.10659146308898926 seconds
number of nodes 399
Max score:4, Next colomn: 2
├── Maximizing node, score: 4
│   ├── Minimizing node, score: 0
│   │   ├── Maximizing node, score: 2
│   │   ├── Maximizing node,score: -inf
│   │   ├── Maximizing node,score: 0
│   │   ├── Maximizing node,score: 0
│   │   ├── Maximizing node,score: 2
│   │   ├── Maximizing node,score: 2
│   │   └── Maximizing node,score: 2
│   ├── Minimizing node, score: 0
│   │   ├── Maximizing node, score: 2
│   │   ├── Maximizing node,score: -inf
│   │   ├── Maximizing node,score: 0
│   │   ├── Maximizing node,score: 0
│   │   ├── Maximizing node,score: 2
│   │   ├── Maximizing node,score: 2
│   │   └── Maximizing node,score: 2
│   ├── Minimizing node, score: 0
│   │   ├── Maximizing node, score: 2
│   │   ├── Maximizing node,score: -inf
```



```
Runtime of minmax = 0.1126565933227539 seconds
number of nodes 399
Max score:2, Next colomn: 2
├── Maximizing node, score: 2
│   ├── Minimizing node, score: 0
│   │   ├── Maximizing node, score: 0
│   │   ├── Maximizing node,score: -inf
│   │   ├── Maximizing node,score: 0
│   │   ├── Maximizing node,score: 0
│   │   ├── Maximizing node,score: 0
│   │   ├── Maximizing node,score: 0
│   │   └── Maximizing node,score: 0
│   ├── Minimizing node, score: 2
│   │   ├── Maximizing node, score: 2
│   │   ├── Maximizing node,score: -inf
│   │   ├── Maximizing node,score: 0
│   │   ├── Maximizing node,score: 0
│   │   ├── Maximizing node,score: 0
│   │   ├── Maximizing node,score: 0
│   │   └── Maximizing node,score: 0
│   ├── Minimizing node, score: 2
│   │   ├── Maximizing node, score: 2
│   │   ├── Maximizing node,score: -inf
```

ages installed successfully: Installed packages: 'numpy' (31 minutes ago)

Depth 5:

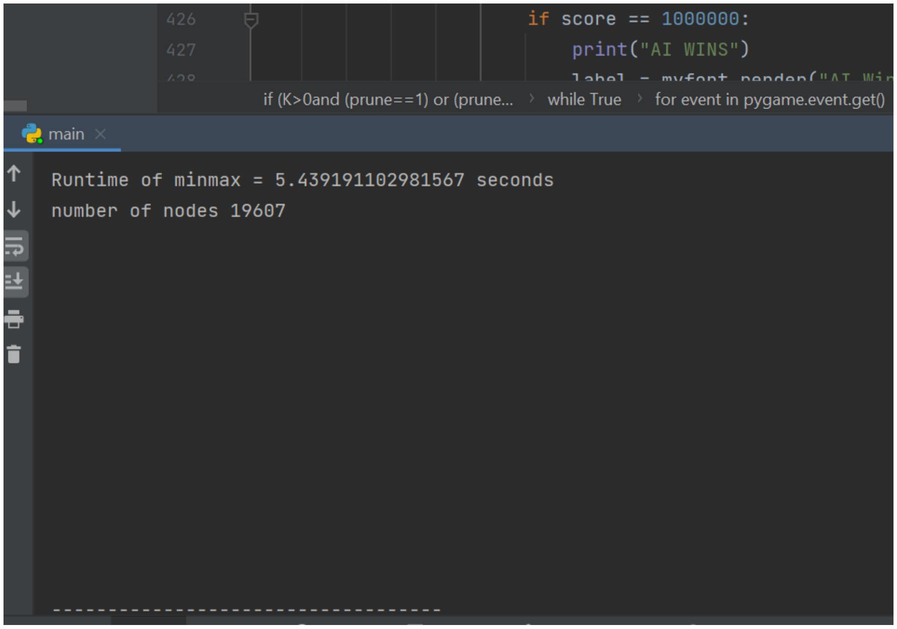


```python
            print(f"number of nodes {number_of_nodes}")
            stringg = f"Max score:{score}, Next colomn: {
            search_tree.update_node(0, tag=stringg)
            # search_tree.show()
            if nextcol == None:
                if score == 1000000:
                    print("AI WINS")
                    label = myfont.render("AI Wins !!!"
```

if (K>0and (prune==1) or (prune... › while True › for event in pygame.event.get() › if event.t

```
|   || 1 ||   || 2 || 2 ||   ||   |
-----------------------------------
Runtime of minmax = 5.70606255531311 seconds
number of nodes 19607
```

Packages installed successfully: Installed packages: 'numpy' (24 minutes ago)



```python
                if score == 1000000:
                    print("AI WINS")
                    label = myfont.render("AI Wi
```

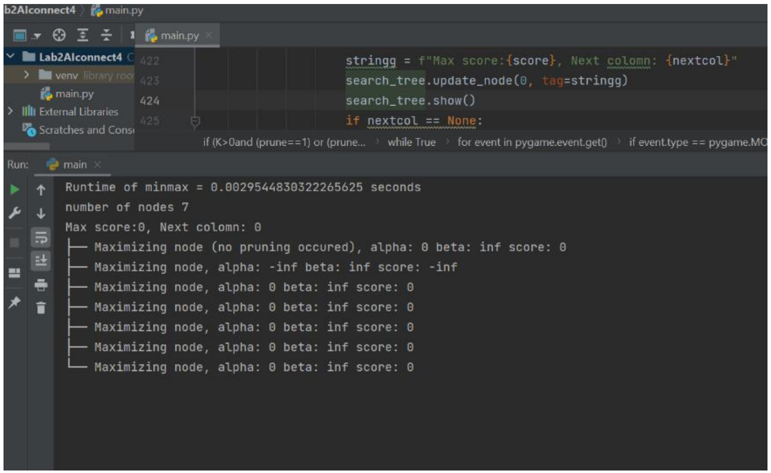
if (K>0and (prune==1) or (prune... › while True › for event in pygame.event.get()

```
Runtime of minmax = 5.439191102981567 seconds
number of nodes 19607




-----------------------------------
```

As shown the time taken by minmax increases as the depth increases because the number of nodes searched increase exponentionally

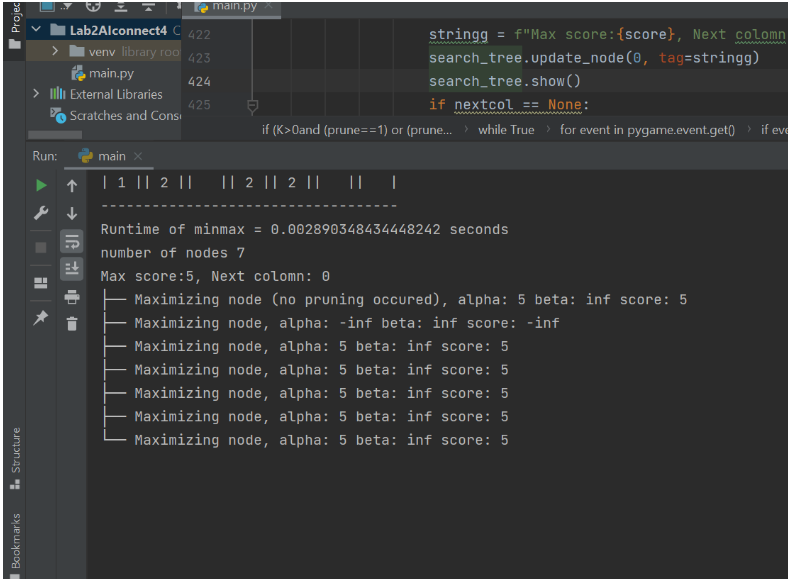
With alpha beta pruning:

Depth 1



```
stringg = f"Max score:{score}, Next colomn: {nextcol}"
search_tree.update_node(0, tag=stringg)
search_tree.show()
if nextcol == None:
```

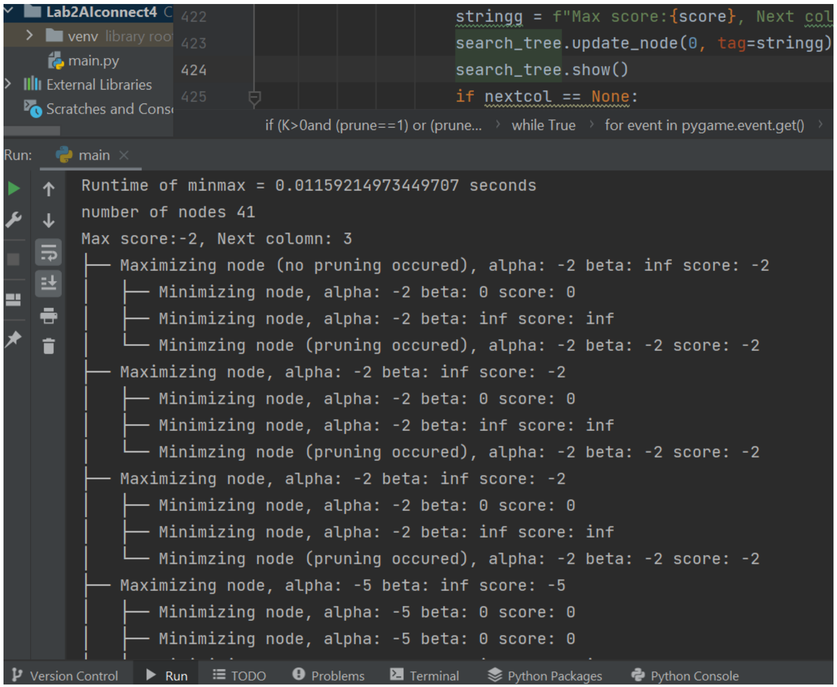
```
Runtime of minmax = 0.0029544830322265625 seconds
number of nodes 7
Max score:0, Next colomn: 0
├── Maximizing node (no pruning occured), alpha: 0 beta: inf score: 0
├── Maximizing node, alpha: -inf beta: inf score: -inf
├── Maximizing node, alpha: 0 beta: inf score: 0
├── Maximizing node, alpha: 0 beta: inf score: 0
├── Maximizing node, alpha: 0 beta: inf score: 0
├── Maximizing node, alpha: 0 beta: inf score: 0
└── Maximizing node, alpha: 0 beta: inf score: 0
```



```
stringg = f"Max score:{score}, Next colomn
search_tree.update_node(0, tag=stringg)
search_tree.show()
if nextcol == None:
```

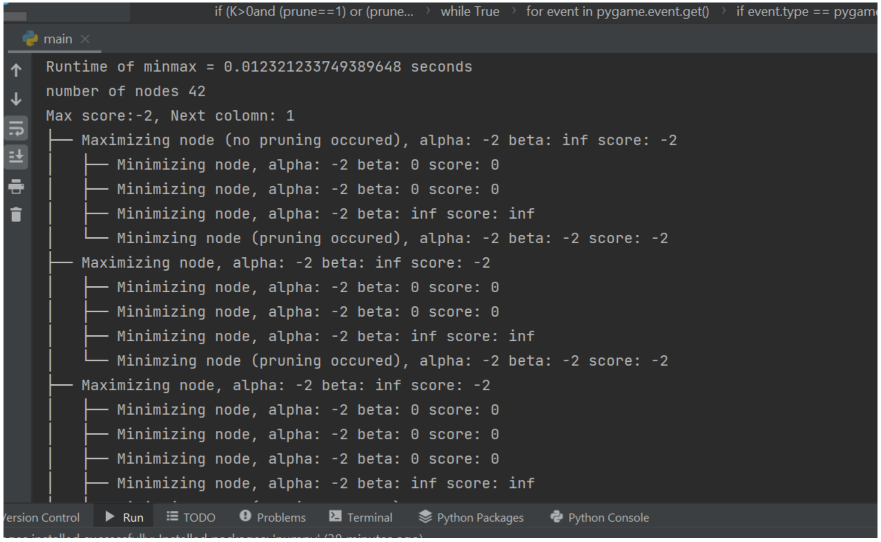
```
| 1 || 2 ||   || 2 || 2 ||   ||   |
-----------------------------------
Runtime of minmax = 0.002890348434448242 seconds
number of nodes 7
Max score:5, Next colomn: 0
├── Maximizing node (no pruning occured), alpha: 5 beta: inf score: 5
├── Maximizing node, alpha: -inf beta: inf score: -inf
├── Maximizing node, alpha: 5 beta: inf score: 5
├── Maximizing node, alpha: 5 beta: inf score: 5
├── Maximizing node, alpha: 5 beta: inf score: 5
├── Maximizing node, alpha: 5 beta: inf score: 5
└── Maximizing node, alpha: 5 beta: inf score: 5
```

Depth 2:



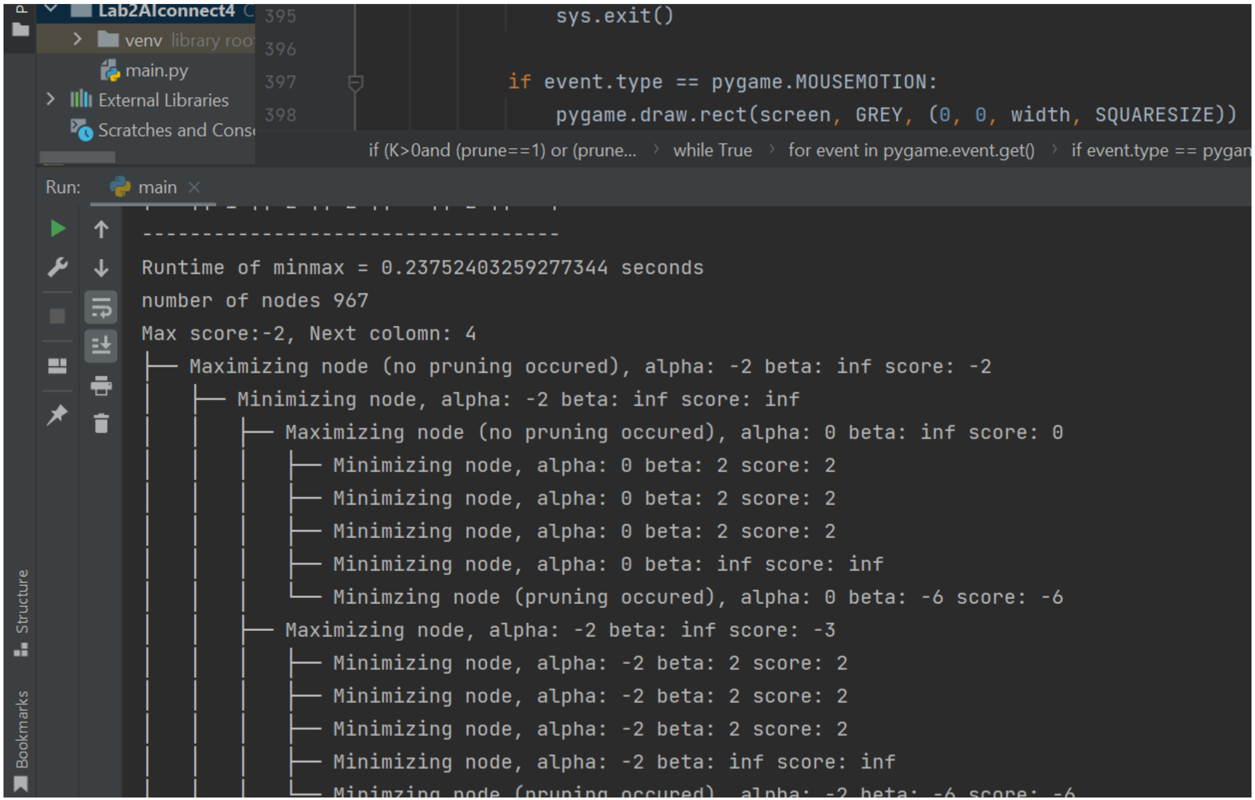
```
            stringg = f"Max score:{score}, Next col
            search_tree.update_node(0, tag=stringg)
            search_tree.show()
            if nextcol == None:
```

```
Runtime of minmax = 0.01159214973449707 seconds
number of nodes 41
Max score:-2, Next colomn: 3
├── Maximizing node (no pruning occured), alpha: -2 beta: inf score: -2
│   ├── Minimizing node, alpha: -2 beta: 0 score: 0
│   ├── Minimizing node, alpha: -2 beta: inf score: inf
│   └── Minimzing node (pruning occured), alpha: -2 beta: -2 score: -2
├── Maximizing node, alpha: -2 beta: inf score: -2
│   ├── Minimizing node, alpha: -2 beta: 0 score: 0
│   ├── Minimizing node, alpha: -2 beta: inf score: inf
│   └── Minimzing node (pruning occured), alpha: -2 beta: -2 score: -2
├── Maximizing node, alpha: -2 beta: inf score: -2
│   ├── Minimizing node, alpha: -2 beta: 0 score: 0
│   ├── Minimizing node, alpha: -2 beta: inf score: inf
│   └── Minimzing node (pruning occured), alpha: -2 beta: -2 score: -2
├── Maximizing node, alpha: -5 beta: inf score: -5
│   ├── Minimizing node, alpha: -5 beta: 0 score: 0
│   ├── Minimizing node, alpha: -5 beta: 0 score: 0
```


if (K>0and (prune==1) or (prune... › while True › for event in pygame.event.get() › if event.type == pygame
main
ersion Control
Run
TODO
Problems
Terminal
Python Packages
Python Console

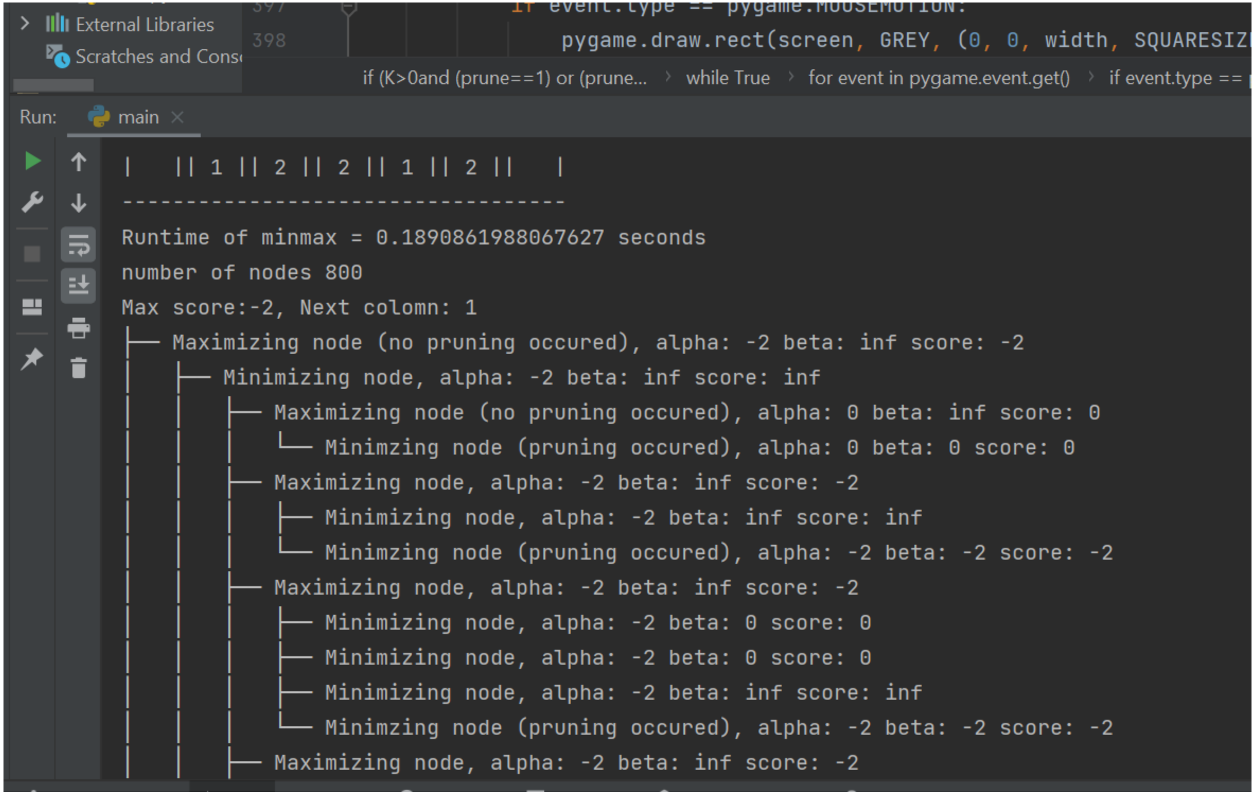
```
Runtime of minmax = 0.012321233749389648 seconds
number of nodes 42
Max score:-2, Next colomn: 1
├── Maximizing node (no pruning occured), alpha: -2 beta: inf score: -2
│   ├── Minimizing node, alpha: -2 beta: 0 score: 0
│   ├── Minimizing node, alpha: -2 beta: 0 score: 0
│   ├── Minimizing node, alpha: -2 beta: inf score: inf
│   └── Minimzing node (pruning occured), alpha: -2 beta: -2 score: -2
├── Maximizing node, alpha: -2 beta: inf score: -2
│   ├── Minimizing node, alpha: -2 beta: 0 score: 0
│   ├── Minimizing node, alpha: -2 beta: 0 score: 0
│   ├── Minimizing node, alpha: -2 beta: inf score: inf
│   └── Minimzing node (pruning occured), alpha: -2 beta: -2 score: -2
├── Maximizing node, alpha: -2 beta: inf score: -2
│   ├── Minimizing node, alpha: -2 beta: 0 score: 0
│   ├── Minimizing node, alpha: -2 beta: 0 score: 0
│   ├── Minimizing node, alpha: -2 beta: 0 score: 0
│   ├── Minimizing node, alpha: -2 beta: inf score: inf
```

Depth 4


Lab2AIconnect4
venv library roo
main.py
External Libraries
Scratches and Cons
Structure
Bookmarks
if (K>0and (prune==1) or (prune... › while True › for event in pygame.event.get() › if event.type == pygam
Run: main

```
            sys.exit()

        if event.type == pygame.MOUSEMOTION:
            pygame.draw.rect(screen, GREY, (0, 0, width, SQUARESIZE))
```

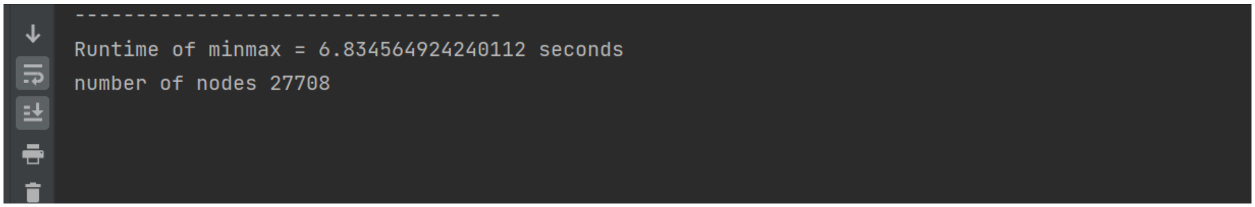
```
----------------------------------
Runtime of minmax = 0.23752403259277344 seconds
number of nodes 967
Max score:-2, Next colomn: 4
├── Maximizing node (no pruning occured), alpha: -2 beta: inf score: -2
│   ├── Minimizing node, alpha: -2 beta: inf score: inf
│   │   ├── Maximizing node (no pruning occured), alpha: 0 beta: inf score: 0
│   │   │   ├── Minimizing node, alpha: 0 beta: 2 score: 2
│   │   │   ├── Minimizing node, alpha: 0 beta: 2 score: 2
│   │   │   ├── Minimizing node, alpha: 0 beta: 2 score: 2
│   │   │   ├── Minimizing node, alpha: 0 beta: inf score: inf
│   │   │   └── Minimzing node (pruning occured), alpha: 0 beta: -6 score: -6
│   │   ├── Maximizing node, alpha: -2 beta: inf score: -3
│   │   │   ├── Minimizing node, alpha: -2 beta: 2 score: 2
│   │   │   ├── Minimizing node, alpha: -2 beta: 2 score: 2
│   │   │   ├── Minimizing node, alpha: -2 beta: 2 score: 2
│   │   │   ├── Minimizing node, alpha: -2 beta: inf score: inf
│   │   │   └── Minimzing node (pruning occured), alpha: -2 beta: -6 score: -6
```


External Libraries
Scratches and Cons
if (K>0and (prune==1) or (prune... › while True › for event in pygame.event.get() › if event.type ==
Run: main

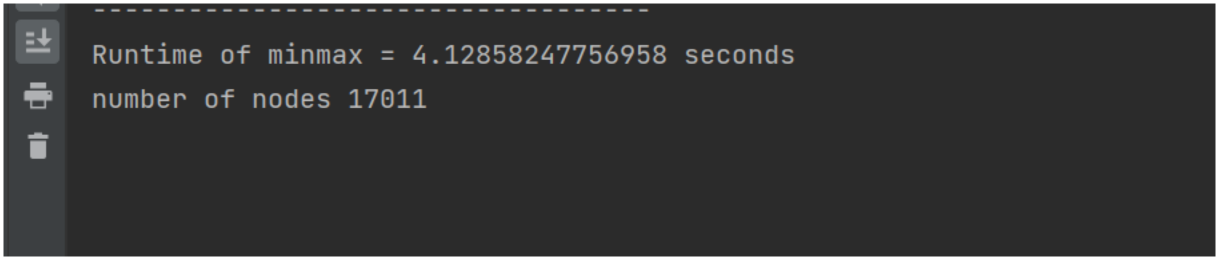
```
        if event.type == pygame.MOUSEMOTION:
            pygame.draw.rect(screen, GREY, (0, 0, width, SQUARESIZE
```

```
|   || 1 || 2 || 2 || 1 || 2 ||   |
----------------------------------
Runtime of minmax = 0.1890861988067627 seconds
number of nodes 800
Max score:-2, Next colomn: 1
├── Maximizing node (no pruning occured), alpha: -2 beta: inf score: -2
│   ├── Minimizing node, alpha: -2 beta: inf score: inf
│   │   ├── Maximizing node (no pruning occured), alpha: 0 beta: inf score: 0
│   │   │   └── Minimzing node (pruning occured), alpha: 0 beta: 0 score: 0
│   │   ├── Maximizing node, alpha: -2 beta: inf score: -2
│   │   │   ├── Minimizing node, alpha: -2 beta: inf score: inf
│   │   │   └── Minimzing node (pruning occured), alpha: -2 beta: -2 score: -2
│   │   ├── Maximizing node, alpha: -2 beta: inf score: -2
│   │   │   ├── Minimizing node, alpha: -2 beta: 0 score: 0
│   │   │   ├── Minimizing node, alpha: -2 beta: 0 score: 0
│   │   │   ├── Minimizing node, alpha: -2 beta: inf score: inf
│   │   │   └── Minimzing node (pruning occured), alpha: -2 beta: -2 score: -2
│   │   ├── Maximizing node, alpha: -2 beta: inf score: -2
```

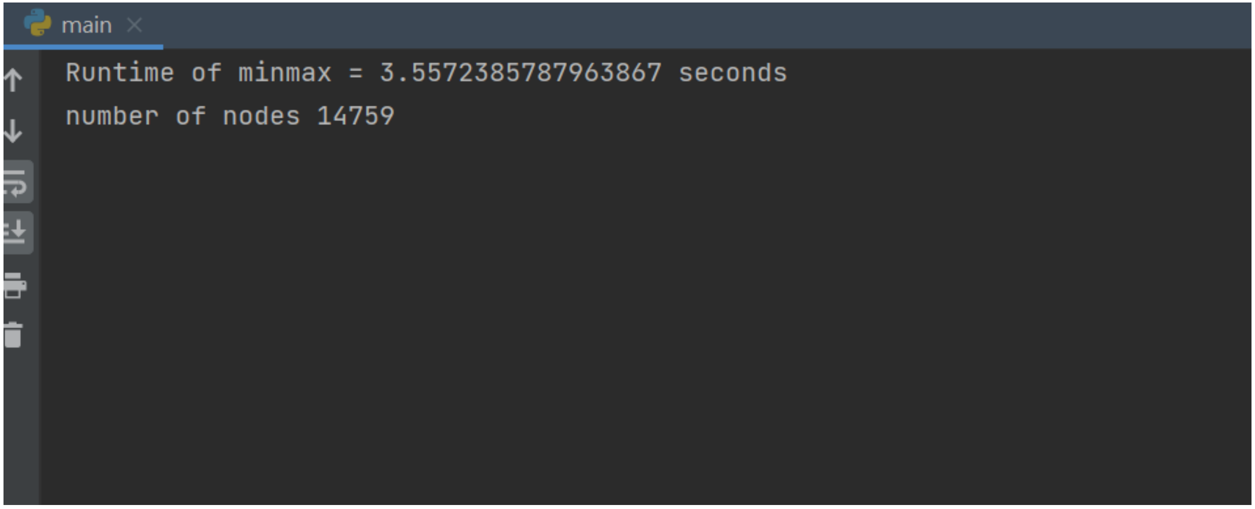
Depth 6:



```
-----------------------------------
Runtime of minmax = 6.834564924240112 seconds
number of nodes 27708
```



```
-----------------------------------
Runtime of minmax = 4.12858247756958 seconds
number of nodes 17011
```



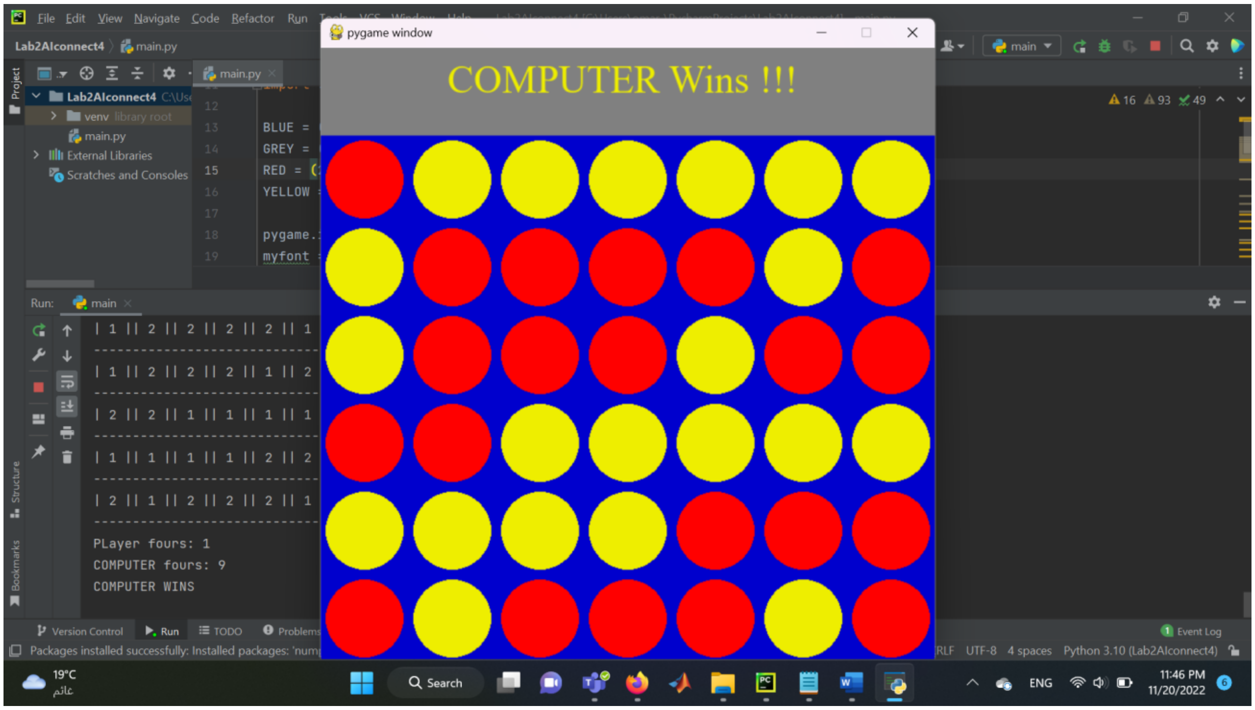
```
Runtime of minmax = 3.5572385787963867 seconds
number of nodes 14759
```

As show the time increases with depth increasing but when we increase the depth

But on average even with depth 6 using alpha beta pruning will be better than using depth 5 without alpha beta because the alpha beta doesn’t always have to search the whole tree

Here are two full games first is using alpha beta and the other is without

And depth 4

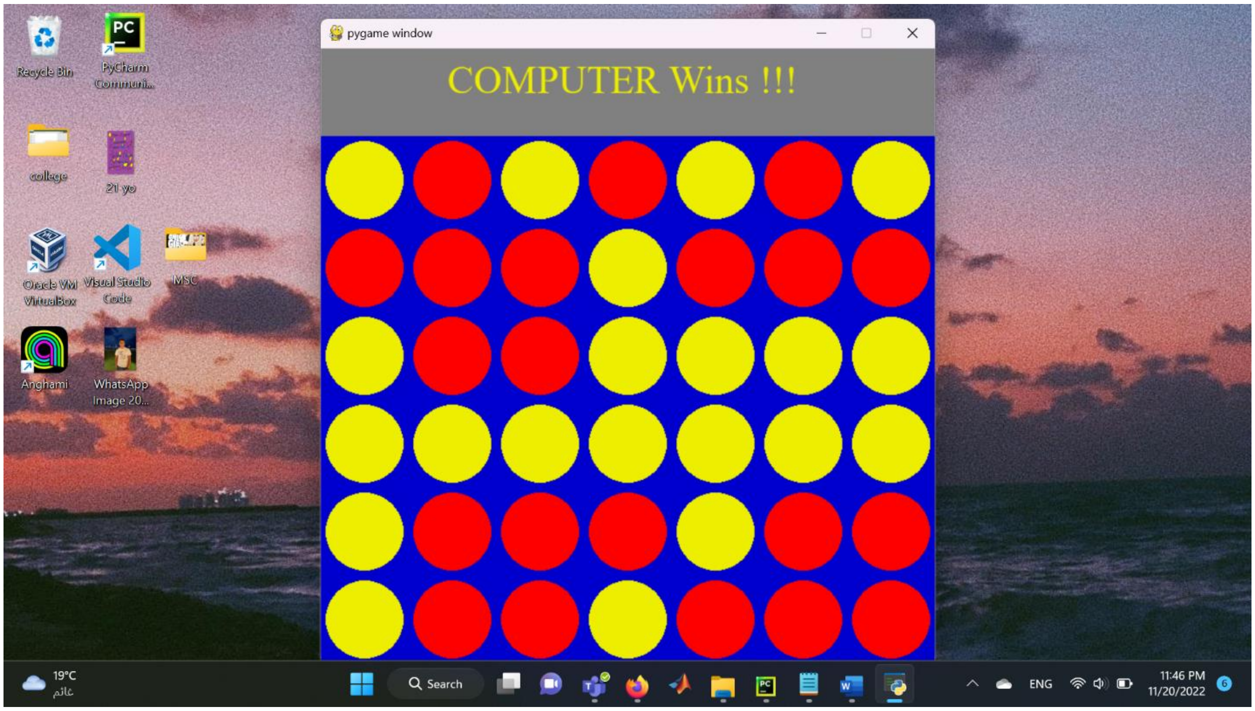


COMPUTER Wins !!!

```
BLUE =
GREY =
RED = (
YELLOW =

pygame.
myfont =
```

```
| 1 || 2 || 2 || 2 || 2 || 1
------------------------------
| 1 || 2 || 2 || 2 || 1 || 2
------------------------------
| 2 || 2 || 1 || 1 || 1 || 1
------------------------------
| 1 || 1 || 1 || 1 || 2 || 2
------------------------------
| 2 || 1 || 2 || 2 || 2 || 1
------------------------------
PLayer fours: 1
COMPUTER fours: 9
COMPUTER WINS
```

Packages installed successfully: Installed packages: 'num



COMPUTER Wins !!!

Node expanded in different depths in above testcases

Depth	Alpha beta	No alpha beta
1	7 nodes and 0.029 secs	7 nodes and 0.0026 secs
2	41 nodes and 0.01 secs	56 nodes and 0.014 secs
3	119 nodes and 0.03 secs	399 nodes an 0.1 secs
4	976 nodes and 0.2 secs	2800 nodes and 0.83 secs
5	2315 nodes and 0.613 secs	19607 nodes and 5.4 secs
6	17011 nodes and 4.1 secs	137255 nodes and 41.25 secs