TITLE OF PROJECT

MINESWEEPER

END TERM REPORT

By

P. Sahith Arya Charan ,

B .Mohan Sai ,

E .Vishnu Vardhan Reddy.

Section: K19PV

Roll Numbers: 61 , 37 , 66

Department of Intelligent Systems,
School of Computer Science Engineering,
Lovely Professional University, Jalandhar
November, 2020

Student Declaration

This is to declare that this report has been written by ~~me~~ us. . No part of the report is copied from other Source. All information included from other sources have been duly acknowledged. We aver that if any Part of the report is found to be Copied we are shall take full responsibility for it.

Mohan.

Mohan Sai

Roll no : 37

Student Declaration

This is to declare that this report has been written by me. No part of the report is copied From other sources. all information included Forom other sources have been duly acknowledged. I aver that if any part of the report is found to be copied. I are shall take full responsibility for it

Signature :- E.vishnu

Name :- Eragamreddy
vishnu vardhan reddy

Roll number :- 66

Student Declaration.

This is to declare that this report has been written by me/us. No part of the report Is copied from other sources. All Information inclutted from other sources have been duly acknowledged. We aver that if any part of the report is found to be Copied, we shall take full responsibility for it.

P. Sahith

P. Sahith Arya Charan

11917559

Gudiwada

31-10-2020

Table of Contents

Title : Mine sweeper

2.1 Game Dynamics -

Minesweeper is a puzzle game which the player will select a cell in a square grid continuously. Each cell hides a bomb or a value which displays the number of bombs in its neighbouring cells. Just to make it clear, neighbouring cells is defined as adjacen horizontally, vertically or diagonally.

2.2 Prerequisites -

1. Understanding of string, int data types.
2. Understanding of list data structures.
3. Understanding of conditional statements.
4. Understanding of for, while loops.
5. Understanding of multidimensional arrays.
6. understanding of declaring and using functions-
7. understanding of using list comprehension.

2.3 Simple steps -

1. Display empty grid
2. Randomly place a bomb
3. Randomly place two bombs within the grid.
4. Levels of difficulty
5. Final Implementation.

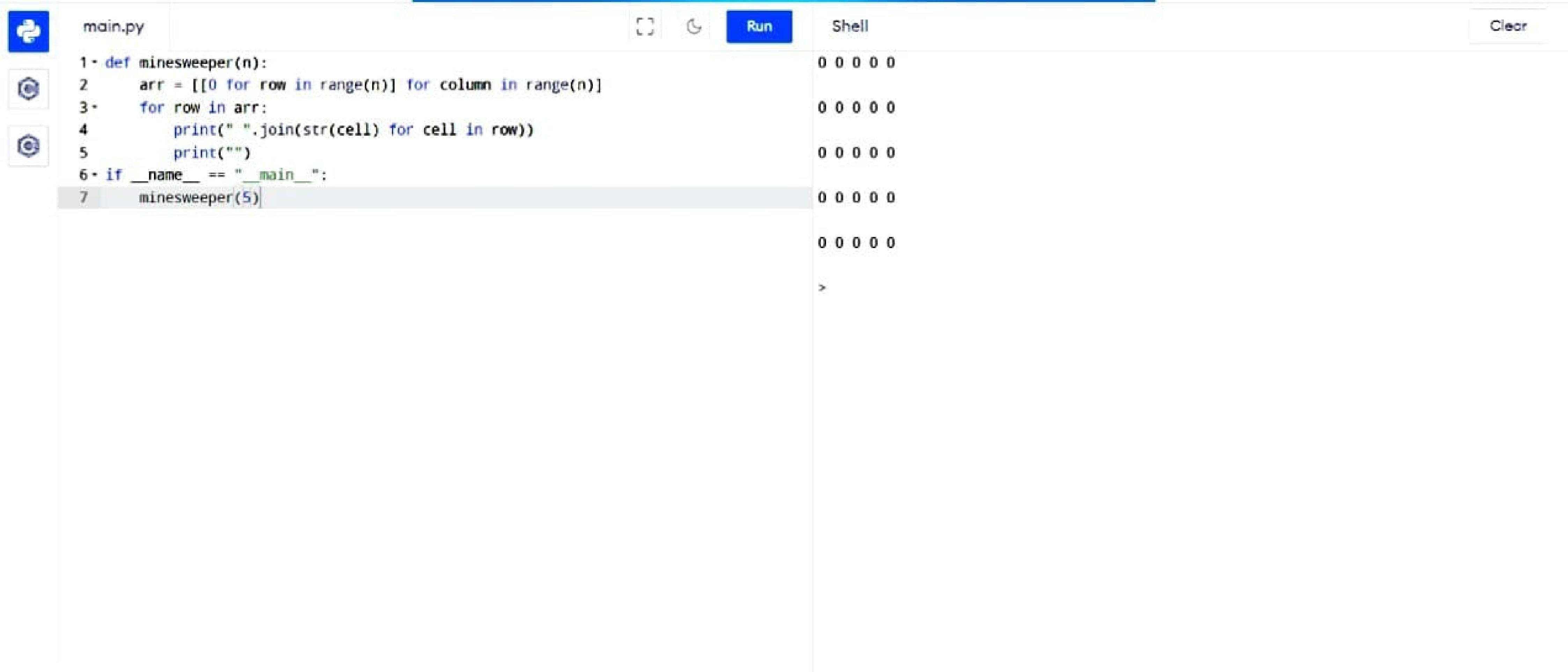
1.1 Display empty grid -

Let's start with creating a empty grid of n rows and n columns

```
def minesweeper (n):
    arr = [[0 for row in range (n)]
    for column in range (n)]
    for row in arr:
        print (" ". Join (str(cell))
    for cell in row))
        print (" ")
if __name__ == "__main__":
    minesweeper (5)
```

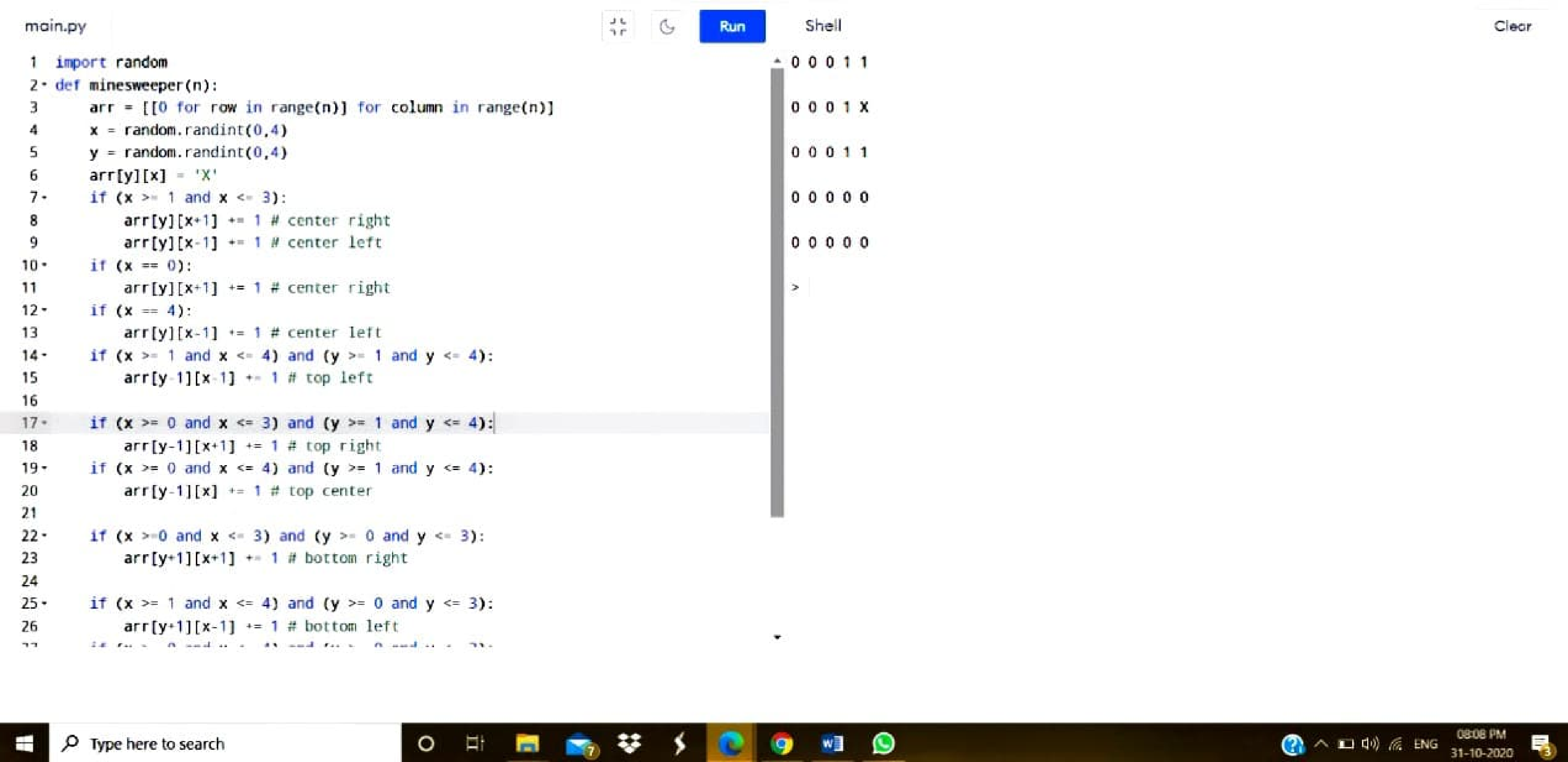
Randomly place a bomb -

Now, let's randomly place a bomb, represented by "X", within the grid. Each of neighbouring cells increases their values to 1.



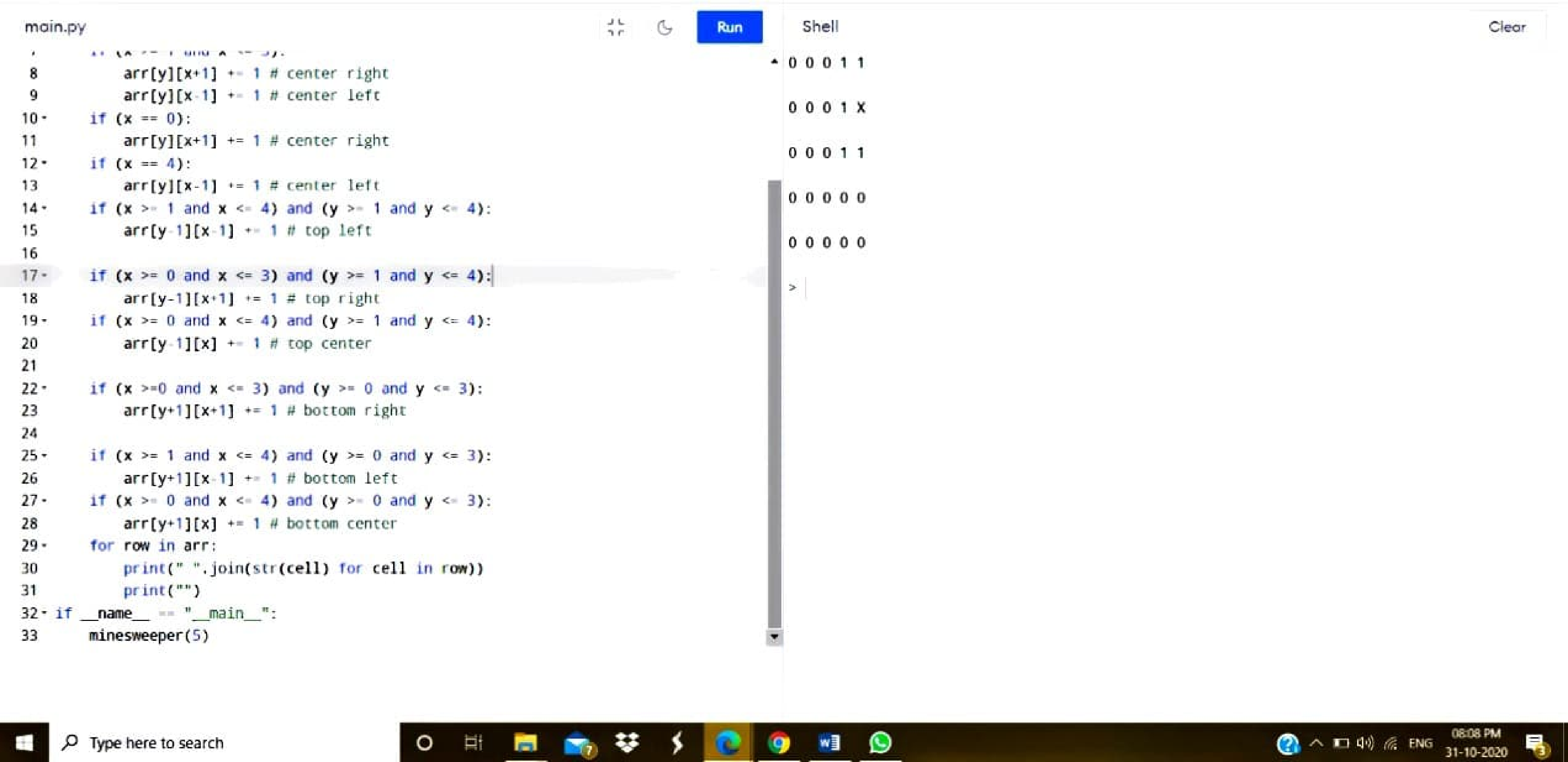
```python
def minesweeper(n):
    arr = [[0 for row in range(n)] for column in range(n)]
    for row in arr:
        print(" ".join(str(cell) for cell in row))
        print("")
if __name__ == "__main__":
    minesweeper(5)
```

```
0 0 0 0 0

0 0 0 0 0

0 0 0 0 0

0 0 0 0 0

0 0 0 0 0

>
```



```python
import random
def minesweeper(n):
    arr = [[0 for row in range(n)] for column in range(n)]
    x = random.randint(0,4)
    y = random.randint(0,4)
    arr[y][x] = 'X'
    if (x >= 1 and x <= 3):
        arr[y][x+1] += 1 # center right
        arr[y][x-1] += 1 # center left
    if (x == 0):
        arr[y][x+1] += 1 # center right
    if (x == 4):
        arr[y][x-1] += 1 # center left
    if (x >= 1 and x <= 4) and (y >= 1 and y <= 4):
        arr[y-1][x-1] += 1 # top left

    if (x >= 0 and x <= 3) and (y >= 1 and y <= 4):
        arr[y-1][x+1] += 1 # top right
    if (x >= 0 and x <= 4) and (y >= 1 and y <= 4):
        arr[y-1][x] += 1 # top center

    if (x >=0 and x <= 3) and (y >= 0 and y <= 3):
        arr[y+1][x+1] += 1 # bottom right

    if (x >= 1 and x <= 4) and (y >= 0 and y <= 3):
        arr[y+1][x-1] += 1 # bottom left
```

```
0 0 0 1 1

0 0 0 1 X

0 0 0 1 1

0 0 0 0 0

0 0 0 0 0

>
```



```python
    if (x >= 1 and x <= 3):
        arr[y][x+1] += 1 # center right
        arr[y][x-1] += 1 # center left
    if (x == 0):
        arr[y][x+1] += 1 # center right
    if (x == 4):
        arr[y][x-1] += 1 # center left
    if (x >= 1 and x <= 4) and (y >= 1 and y <= 4):
        arr[y-1][x-1] += 1 # top left

    if (x >= 0 and x <= 3) and (y >= 1 and y <= 4):
        arr[y-1][x+1] += 1 # top right
    if (x >= 0 and x <= 4) and (y >= 1 and y <= 4):
        arr[y-1][x] += 1 # top center

    if (x >=0 and x <= 3) and (y >= 0 and y <= 3):
        arr[y+1][x+1] += 1 # bottom right

    if (x >= 1 and x <= 4) and (y >= 0 and y <= 3):
        arr[y+1][x-1] += 1 # bottom left
    if (x >= 0 and x <= 4) and (y >= 0 and y <= 3):
        arr[y+1][x] += 1 # bottom center
    for row in arr:
        print(" ".join(str(cell) for cell in row))
        print("")
if __name__ == "__main__":
    minesweeper(5)
```

```
0 0 0 1 1

0 0 0 1 X

0 0 0 1 1

0 0 0 0 0

0 0 0 0 0

>
```

Explaination -

	0	1	2	3	4
0					
1		1	1	1	
2		1	X	1	
3		1	1	1	
4					

Essentially, the bomb should have it's neighbours cells increase to 1:

1. Top left 2. Top center 3. Top Right 4. Center left

5. Center Right 6. Bottom left 7. Bottom Center 8. Bottom Right

However, there might be instances where the bomb would be plac

at extreme corners of the grid.

The following explains the Conditions to take into consideration when increasing the neighboring cell value to 1.

	0	1	2	3	4
0	1	1	1	1	
1	1	×	×	×	×
2	1	×	×	×	×
3	1	×	×	×	×
4		×	×	×	×

Top Left

The condition for placing the bomb (x) at TOP left should be

with in :

1. 1 and 4 cells of the x-axis and

2. 1 and 4 ~~of~~ cells of the y-axis.

If you place the bomb (x) anywhere outside of the box, the TOP left neighbour of the bomb (x) will not be displayed. Also, this is to ensure no out of bound exceptions occuring.

	0	1	2	3	4
0	1	1	1	1	1
1	x	x	x	x	x
2	x	x	x	x	x
3	x	x	x	x	x
4	x	x	x	x	x

Top center

The condition for placing the bomb (x) at Top center should be within :

1. 0 and 4 cell of the x-axis and

2. 1 and 4 cell of the y-axis

If you place the bomb (x) anywhere outside of the box, the Top center neighbour of the bomb (x) with not be displayed. Also, this is to ensure no out of bound exceptions occuring.

	0	1	2	3	4
0		1	1	1	1
1	x	x	x	x	1
2	x	x	x	x	1
3	x	x	x	x	1
4	x	x	x	x	

Top right

The condition for placing the bomb (x) at Top right should be within:

1. 0 and 3 cell of the x-axis and

2. 1 and 4 cell of the y-axis

If you place the bomb (x) anywhere outside of the box, the Top right neighbour of the bomb (x) ~~with~~ will not be displayed. Also, this is to ensure no out of bound exceptions occuring.

	0	1	2	3	4
0	1	x	x	x	x
1	1	x	x	x	x
2	1	x	x	x	x
3	1	x	x	x	x
4	1	x	x	x	x

center Left

The condition for placing the bomb (x) at center left should be within:

1. 1 and 4 cell of the x-axis and 2. 0 and 4 cell of the y axis.

If you place the bomb (x) anywhere outside of the box, the center left neighbour of the bomb (x) will not be displayed.

Also, this is to ensure no out of bound exceptions occuring.

-

	0	1	2	3	4
0	X	X	X	X	1
1	X	X	X	X	1
2	X	X	X	X	1
3	X	X	X	X	1
4	X	X	X	X	1

Center
Right

The Condition for placing the bomb (X) at center right Should be within:

- 0 and 3 cell of the x axis and
- 0 and 4 cell of the y axis

If you place of bomb (X) anywhere outside of the red box, the center right neighbour of the bomb (x) will not be displayed. Also, this is ~~to~~ ensure not out of bound exceptions Occuring.

	0	1	2	3	4
0		X	X	X	X
1	1	X	X	X	X
2	1	X	X	X	X
3	1	X	X	X	X
4	1	1	1	1	

Bottom Left

The condition for placing the bomb (x) at bottom left should be within:

If you place the bomb (x) anywhere out side the box, the bottom left neighbour of the bomb (x) will not be displayed.

Also, this is to ensure no out of bound exceptions occuring.

	0	1	2	3	4
0	X	X	X	X	X
1	X	X	X	X	X
2	X	X	X	X	X
3	X	X	X	X	X
4	1	1	1	1	1

Bottom Center

The condition for placing the bomb (x) at bottom center. Should be within

- 0 and 4 cell of the x axis and
- 0 and 3 cell of the y axis.

If you place the bomb (x) anywhere outside of the box, the bottom center neighbour of the bomb (x) will not be displayed. Also, this is to ensure no out of bound exceptions occuring.

	0	1	2	3	4
0	X	X	X	X	
1	X	X	X	X	1
2	X	X	X	X	1
3	X	X	X	X	1
4		1	1	1	1

Bottom Right

The condition for placing the bomb (X) at Bottom right should be within:

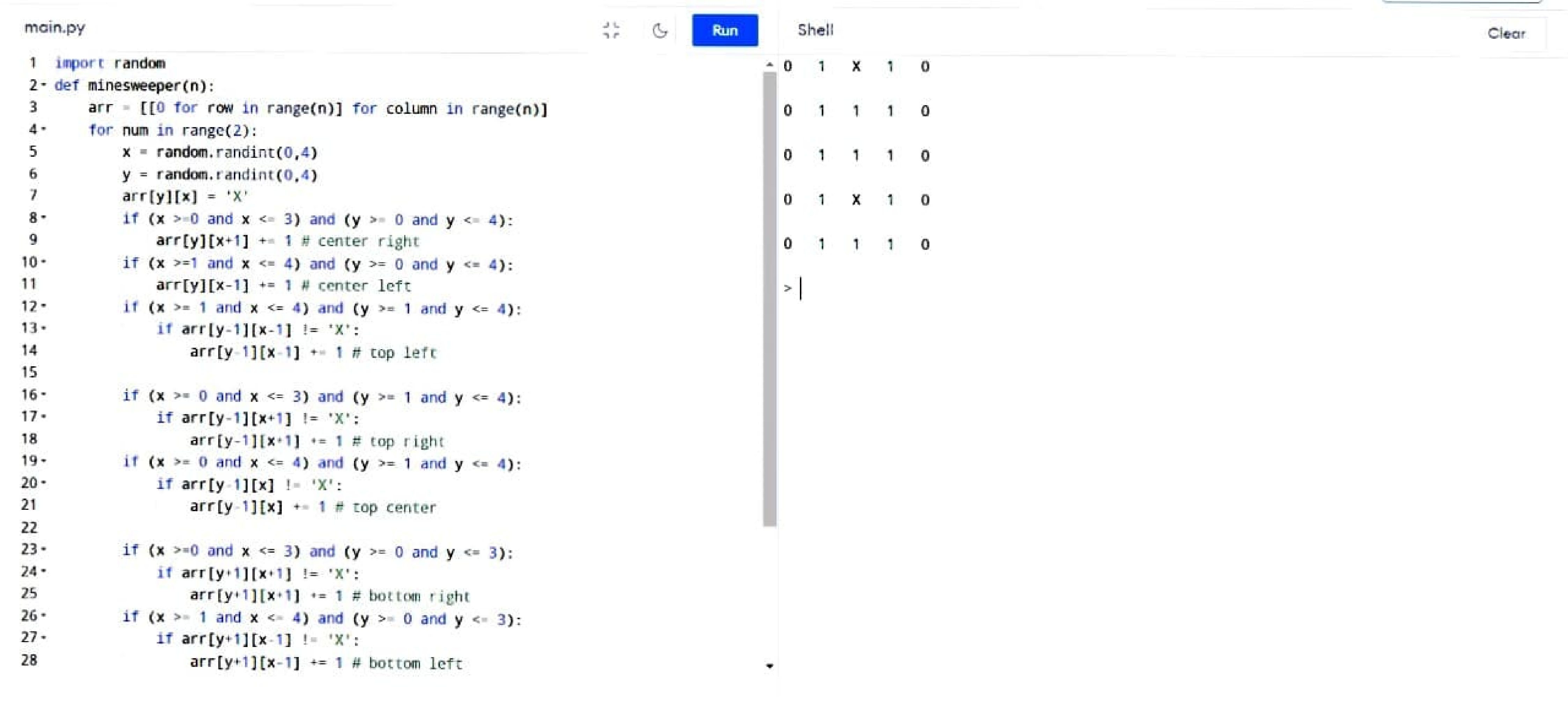
- 0 and 3 cell of the x axis and
- 0 and 3 cell of the y axis

If you place of the bomb (X) anywhere outside the box, the bottom Right neighbour of the bomb (X) will not be displayed. Also, this is to ensure not out of bound expections occuring.

1.2

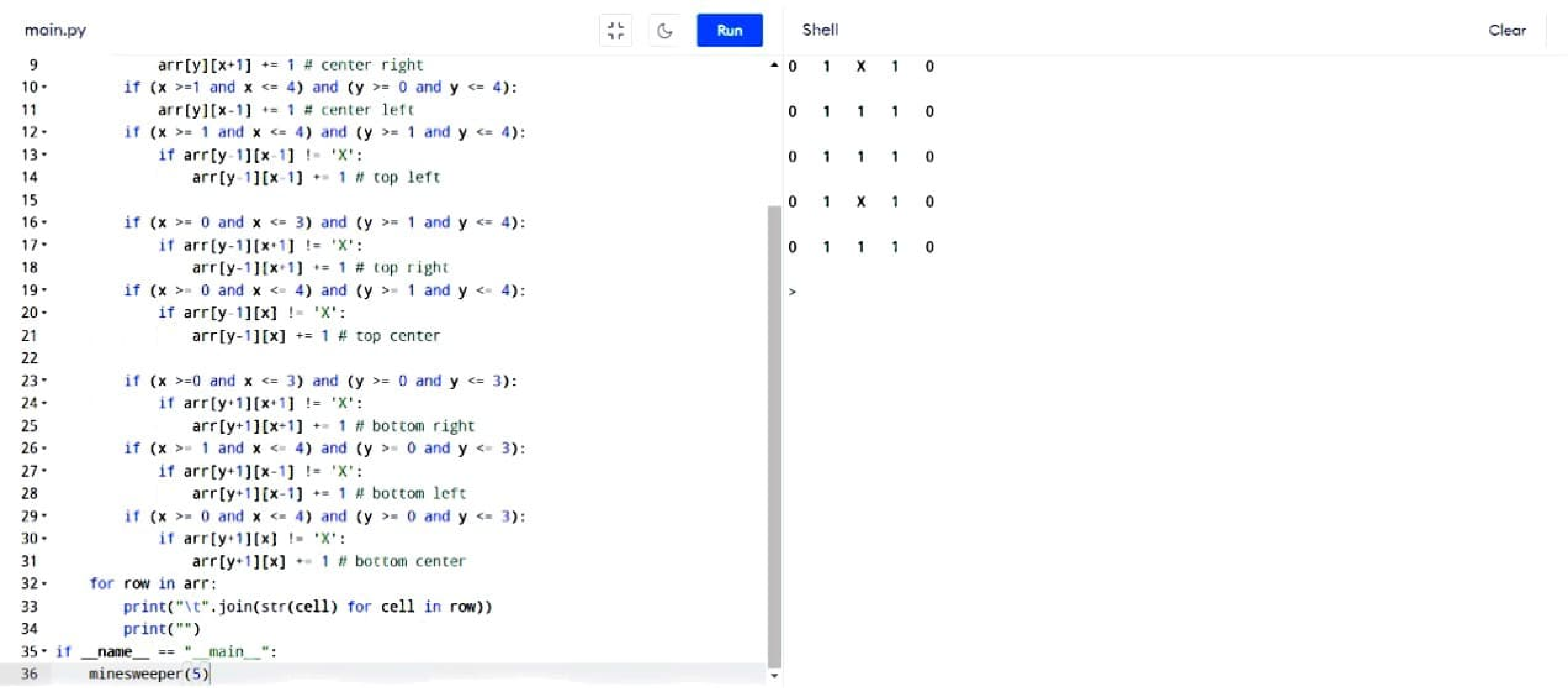
Randomly place two bombs within the grid

Each of neighboring cells to be populated with correct values.



```python
import random
def minesweeper(n):
    arr = [[0 for row in range(n)] for column in range(n)]
    for num in range(2):
        x = random.randint(0,4)
        y = random.randint(0,4)
        arr[y][x] = 'X'
        if (x >=0 and x <= 3) and (y >= 0 and y <= 4):
            arr[y][x+1] += 1 # center right
        if (x >=1 and x <= 4) and (y >= 0 and y <= 4):
            arr[y][x-1] += 1 # center left
        if (x >= 1 and x <= 4) and (y >= 1 and y <= 4):
            if arr[y-1][x-1] != 'X':
                arr[y-1][x-1] += 1 # top left

        if (x >= 0 and x <= 3) and (y >= 1 and y <= 4):
            if arr[y-1][x+1] != 'X':
                arr[y-1][x+1] += 1 # top right
        if (x >= 0 and x <= 4) and (y >= 1 and y <= 4):
            if arr[y-1][x] != 'X':
                arr[y-1][x] += 1 # top center

        if (x >=0 and x <= 3) and (y >= 0 and y <= 3):
            if arr[y+1][x+1] != 'X':
                arr[y+1][x+1] += 1 # bottom right
        if (x >= 1 and x <= 4) and (y >= 0 and y <= 3):
            if arr[y+1][x-1] != 'X':
                arr[y+1][x-1] += 1 # bottom left
```

```
0   1   X   1   0

0   1   1   1   0

0   1   1   1   0

0   1   X   1   0

0   1   1   1   0

>
```



```python
            arr[y][x+1] += 1 # center right
        if (x >=1 and x <= 4) and (y >= 0 and y <= 4):
            arr[y][x-1] += 1 # center left
        if (x >= 1 and x <= 4) and (y >= 1 and y <= 4):
            if arr[y-1][x-1] != 'X':
                arr[y-1][x-1] += 1 # top left

        if (x >= 0 and x <= 3) and (y >= 1 and y <= 4):
            if arr[y-1][x+1] != 'X':
                arr[y-1][x+1] += 1 # top right
        if (x >= 0 and x <= 4) and (y >= 1 and y <= 4):
            if arr[y-1][x] != 'X':
                arr[y-1][x] += 1 # top center

        if (x >=0 and x <= 3) and (y >= 0 and y <= 3):
            if arr[y+1][x+1] != 'X':
                arr[y+1][x+1] += 1 # bottom right
        if (x >= 1 and x <= 4) and (y >= 0 and y <= 3):
            if arr[y+1][x-1] != 'X':
                arr[y+1][x-1] += 1 # bottom left
        if (x >= 0 and x <= 4) and (y >= 0 and y <= 3):
            if arr[y+1][x] != 'X':
                arr[y+1][x] += 1 # bottom center
    for row in arr:
        print("\t".join(str(cell) for cell in row))
        print("")
if __name__ == "__main__":
    minesweeper(5)
```

```
0	1	X	1	0

0	1	1	1	0

0	1	1	1	0

0	1	X	1	0

0	1	1	1	0

>
```

Levels of difficulty

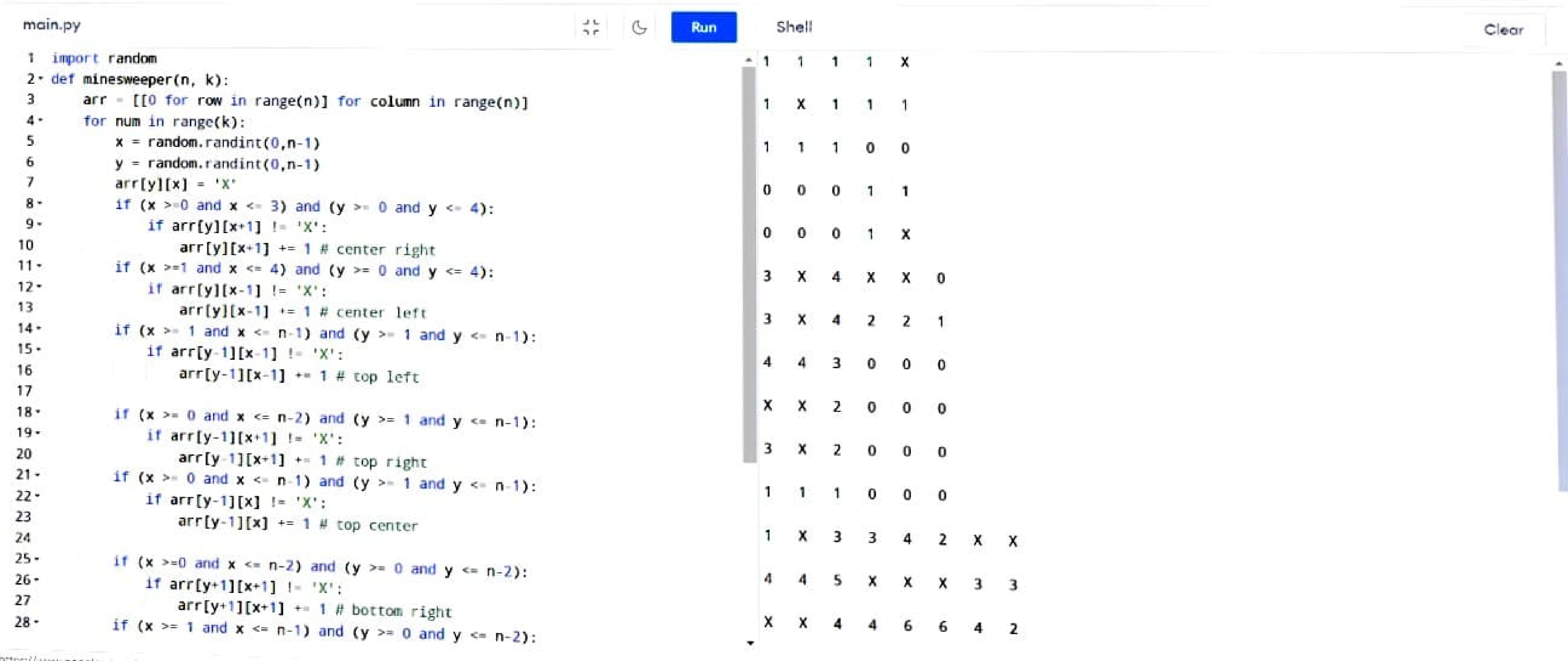
Now, let's generate k numbers of bombs within a grid of size n and correctly display all the values in the neighbouring cells surrounding the bombs.

The following are the levels of difficulty:

Beginner (grid size n = 5; no of bombs k = 3)

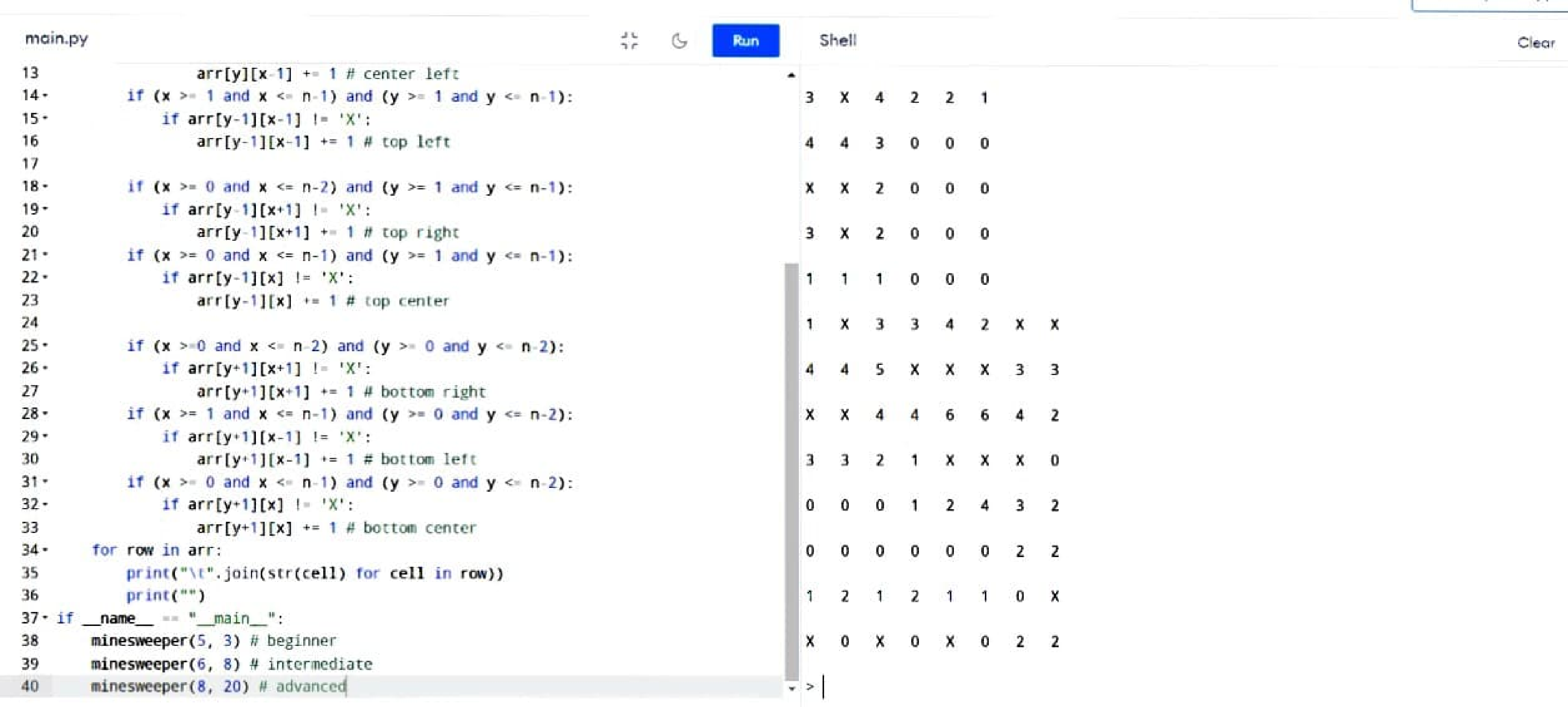
Intermediate (grid size n = 6; no of bombs k = 8

Expert (grid size n = 8; no of bombs k = 20)



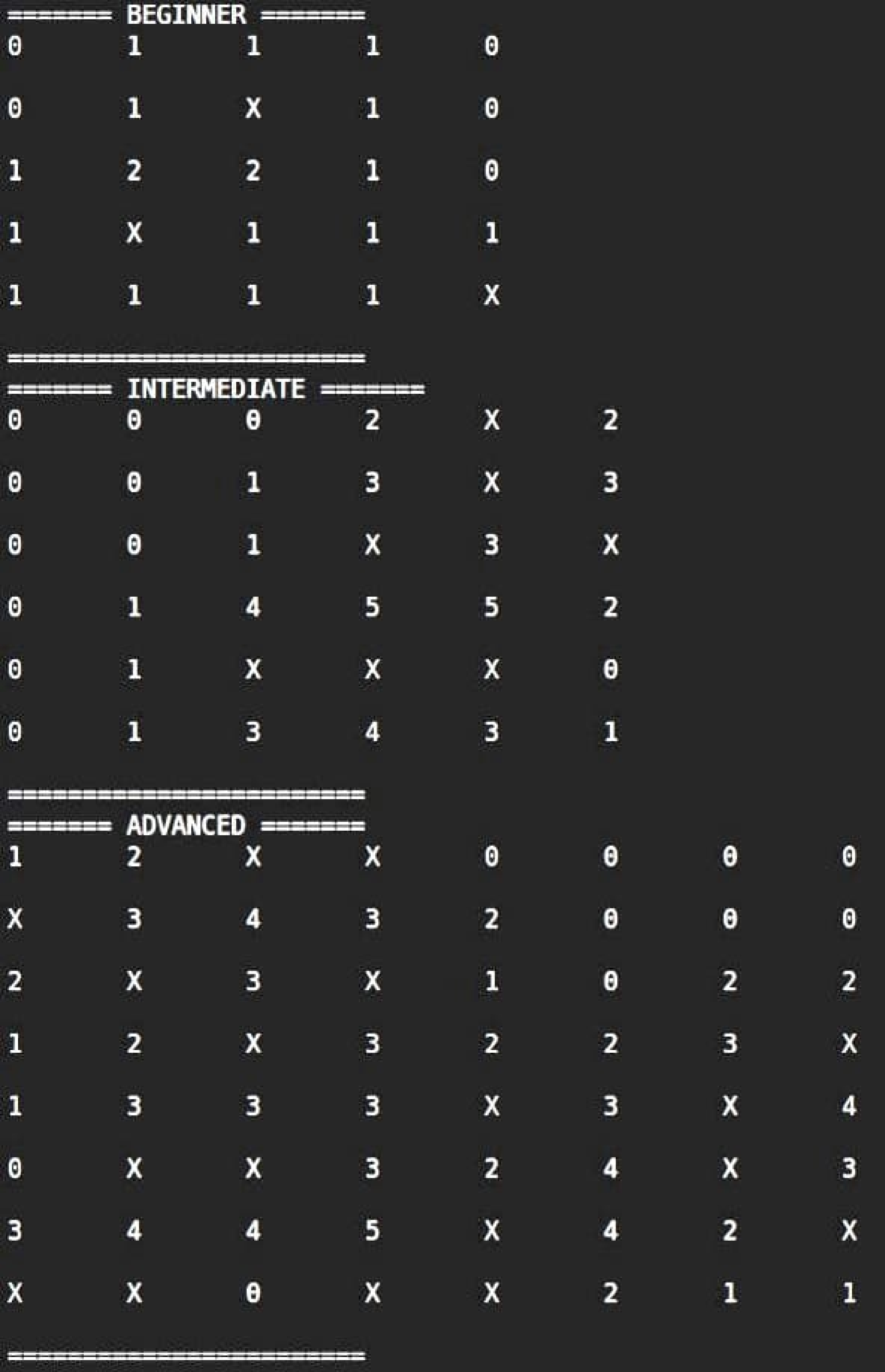
```python
import random
def minesweeper(n, k):
    arr = [[0 for row in range(n)] for column in range(n)]
    for num in range(k):
        x = random.randint(0,n-1)
        y = random.randint(0,n-1)
        arr[y][x] = 'X'
        if (x >=0 and x <= 3) and (y >= 0 and y <= 4):
            if arr[y][x+1] != 'X':
                arr[y][x+1] += 1 # center right
        if (x >=1 and x <= 4) and (y >= 0 and y <= 4):
            if arr[y][x-1] != 'X':
                arr[y][x-1] += 1 # center left
        if (x >= 1 and x <= n-1) and (y >= 1 and y <= n-1):
            if arr[y-1][x-1] != 'X':
                arr[y-1][x-1] += 1 # top left

        if (x >= 0 and x <= n-2) and (y >= 1 and y <= n-1):
            if arr[y-1][x+1] != 'X':
                arr[y-1][x+1] += 1 # top right
        if (x >= 0 and x <= n-1) and (y >= 1 and y <= n-1):
            if arr[y-1][x] != 'X':
                arr[y-1][x] += 1 # top center

        if (x >=0 and x <= n-2) and (y >= 0 and y <= n-2):
            if arr[y+1][x+1] != 'X':
                arr[y+1][x+1] += 1 # bottom right
        if (x >= 1 and x <= n-1) and (y >= 0 and y <= n-2):
```

```
1   1   1   1   X

1   X   1   1   1

1   1   1   0   0

0   0   0   1   1

0   0   0   1   X

3   X   4   X   X   0

3   X   4   2   2   1

4   4   3   0   0   0

X   X   2   0   0   0

3   X   2   0   0   0

1   1   1   0   0   0

1   X   3   3   4   2   X   X

4   4   5   X   X   X   3   3

X   X   4   4   6   6   4   2
```



```python
                arr[y][x-1] += 1 # center left
        if (x >= 1 and x <= n-1) and (y >= 1 and y <= n-1):
            if arr[y-1][x-1] != 'X':
                arr[y-1][x-1] += 1 # top left

        if (x >= 0 and x <= n-2) and (y >= 1 and y <= n-1):
            if arr[y-1][x+1] != 'X':
                arr[y-1][x+1] += 1 # top right
        if (x >= 0 and x <= n-1) and (y >= 1 and y <= n-1):
            if arr[y-1][x] != 'X':
                arr[y-1][x] += 1 # top center

        if (x >=0 and x <= n-2) and (y >= 0 and y <= n-2):
            if arr[y+1][x+1] != 'X':
                arr[y+1][x+1] += 1 # bottom right
        if (x >= 1 and x <= n-1) and (y >= 0 and y <= n-2):
            if arr[y+1][x-1] != 'X':
                arr[y+1][x-1] += 1 # bottom left
        if (x >= 0 and x <= n-1) and (y >= 0 and y <= n-2):
            if arr[y+1][x] != 'X':
                arr[y+1][x] += 1 # bottom center
    for row in arr:
        print("\t".join(str(cell) for cell in row))
        print("")
if __name__ == "__main__":
    minesweeper(5, 3) # beginner
    minesweeper(6, 8) # intermediate
    minesweeper(8, 20) # advanced
```

```
3	X	4	2	2	1

4	4	3	0	0	0

X	X	2	0	0	0

3	X	2	0	0	0

1	1	1	0	0	0

1	X	3	3	4	2	X	X

4	4	5	X	X	X	3	3

X	X	4	4	6	6	4	2

3	3	2	1	X	X	X	0

0	0	0	1	2	4	3	2

0	0	0	0	0	0	2	2

1	2	1	2	1	1	0	X

X	0	X	0	X	0	2	2

> 
```

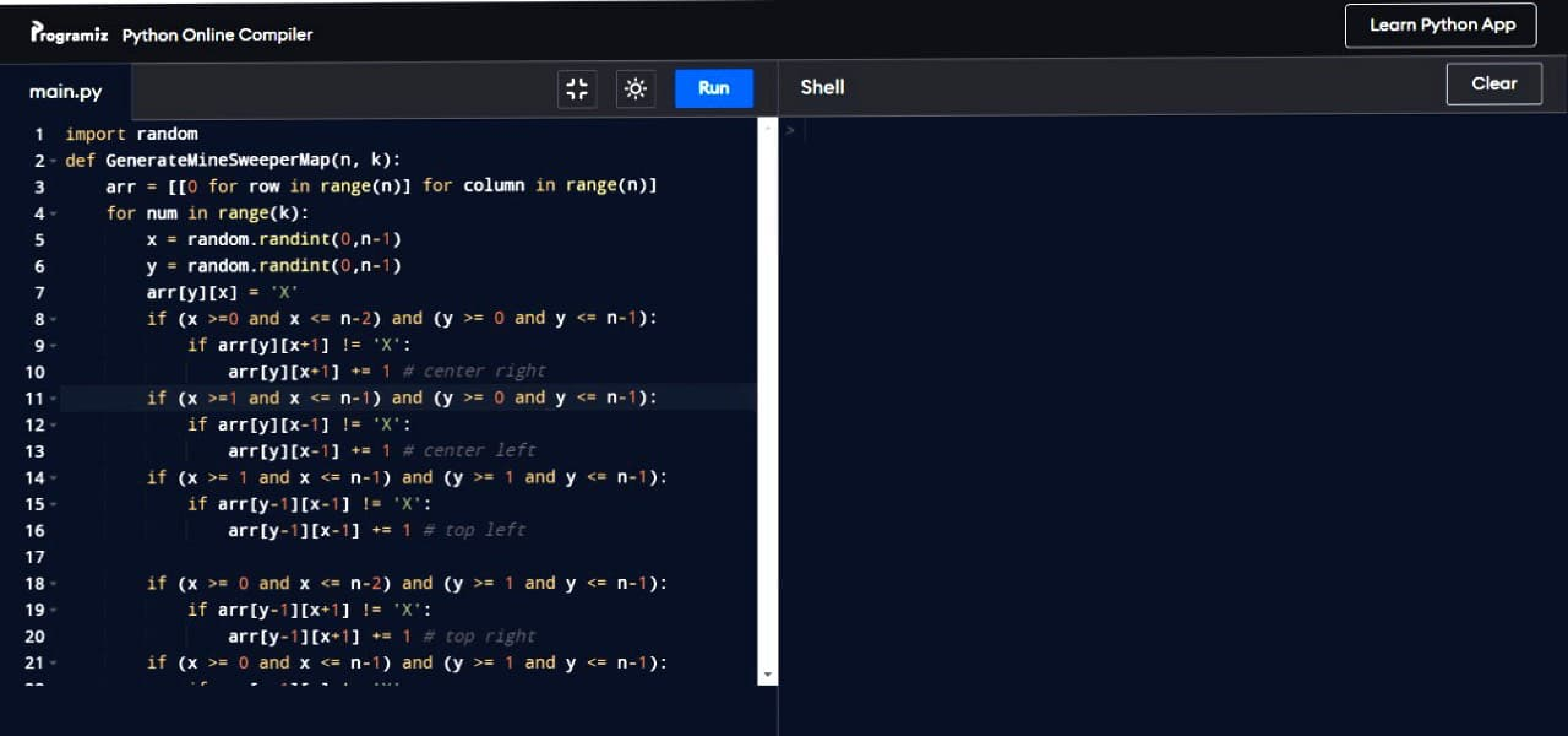


```
======= BEGINNER =======
0       1       1       1       0

0       1       X       1       0

1       2       2       1       0

1       X       1       1       1

1       1       1       1       X

========================
======= INTERMEDIATE =======
0       0       0       2       X       2

0       0       1       3       X       3

0       0       1       X       3       X

0       1       4       5       5       2

0       1       X       X       X       0

0       1       3       4       3       1

========================
======= ADVANCED =======
1       2       X       X       0       0       0       0

X       3       4       3       2       0       0       0

2       X       3       X       1       0       2       2

1       2       X       3       2       2       3       X

1       3       3       3       X       3       X       4

0       X       X       3       2       4       X       3

3       4       4       5       X       4       2       X

X       X       0       X       X       2       1       1

========================
```

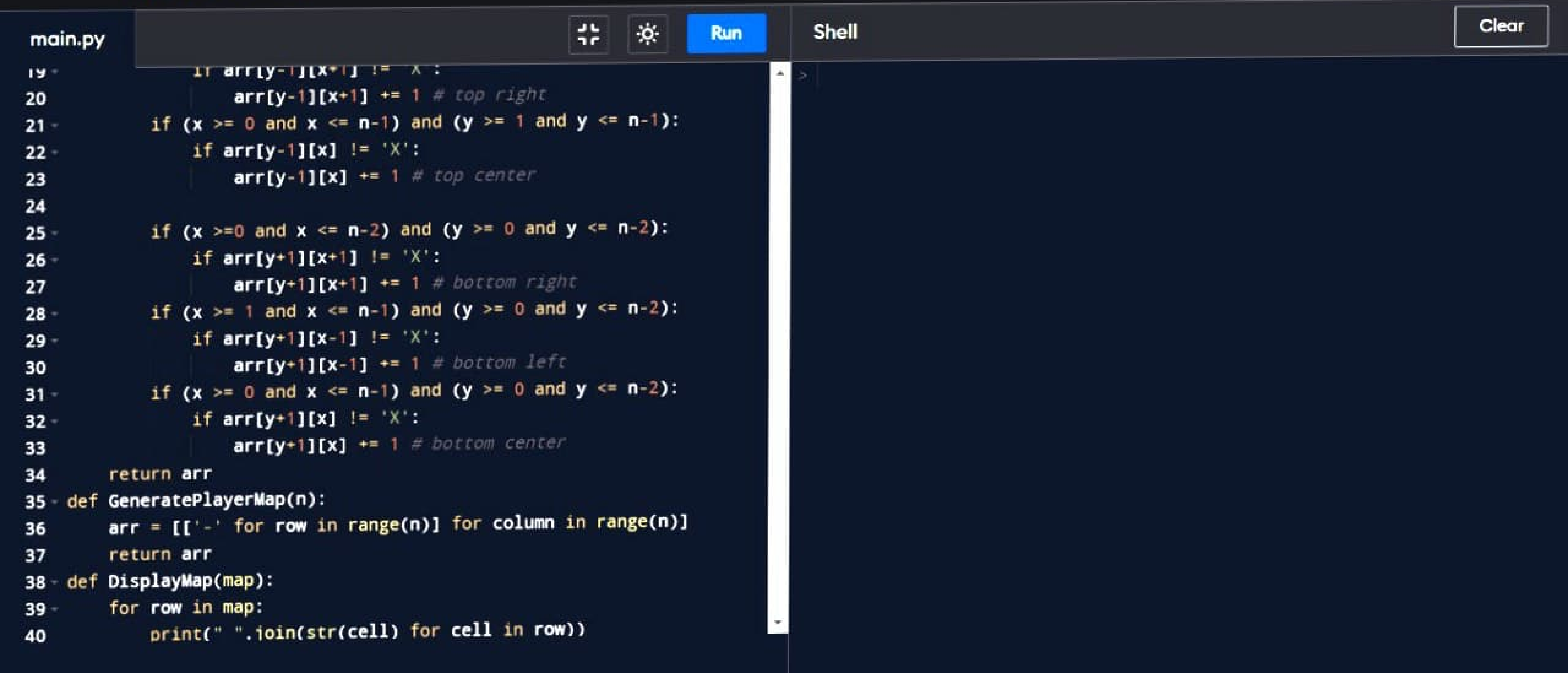
Essentially, our source code

in consideration the dynamic size of the grid and number of bombs. Therefore, instead of hardcoding a fixed size, we replace size n into the conditional statements where necessary

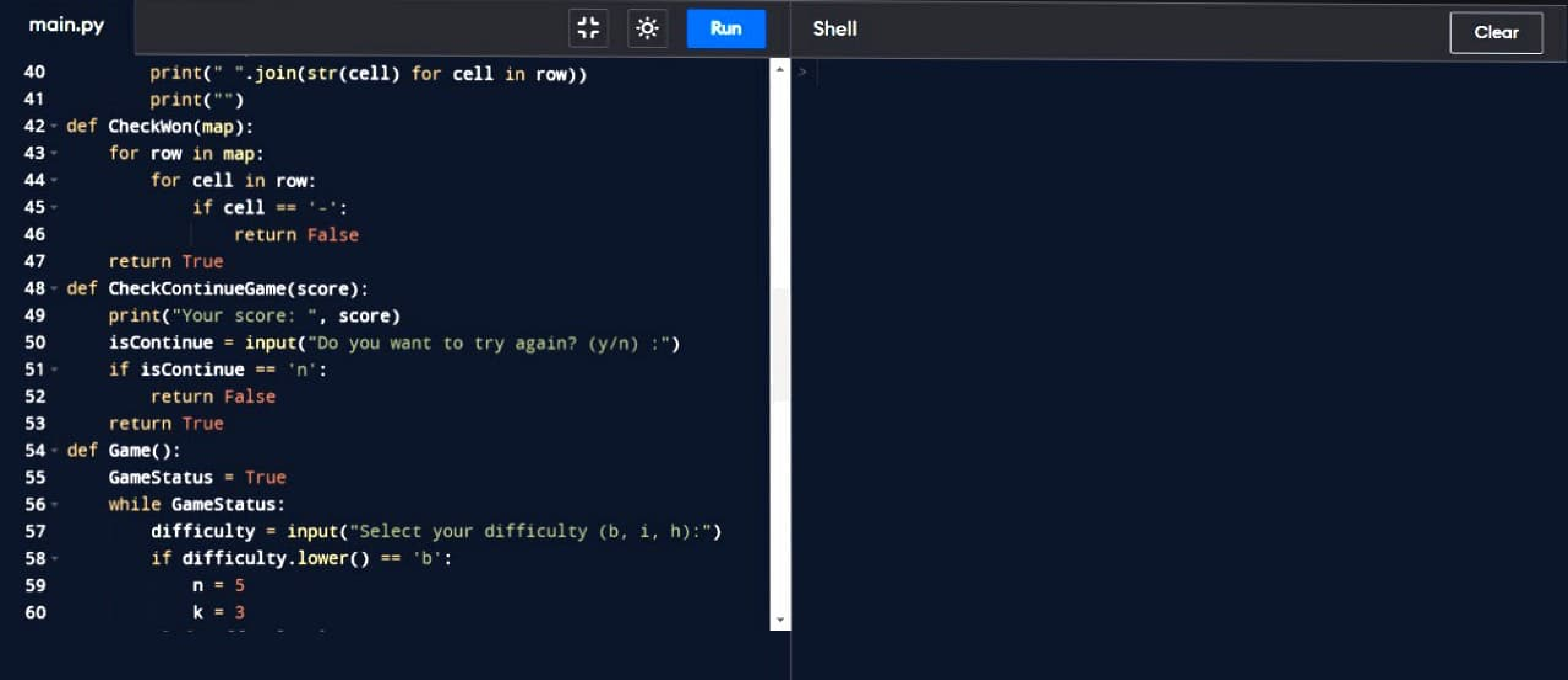
1.3 - Final implemention of game



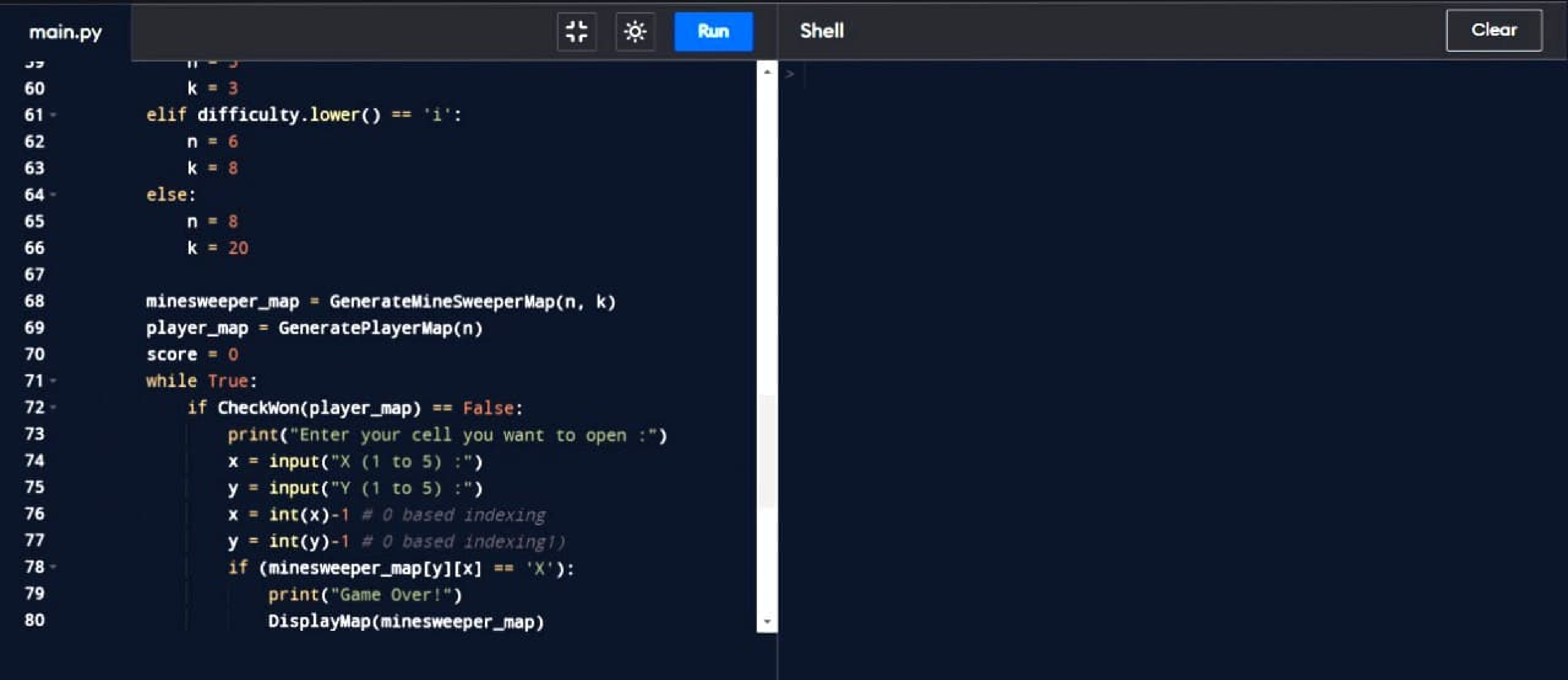
```python
import random
def GenerateMineSweeperMap(n, k):
    arr = [[0 for row in range(n)] for column in range(n)]
    for num in range(k):
        x = random.randint(0,n-1)
        y = random.randint(0,n-1)
        arr[y][x] = 'X'
        if (x >=0 and x <= n-2) and (y >= 0 and y <= n-1):
            if arr[y][x+1] != 'X':
                arr[y][x+1] += 1 # center right
        if (x >=1 and x <= n-1) and (y >= 0 and y <= n-1):
            if arr[y][x-1] != 'X':
                arr[y][x-1] += 1 # center left
        if (x >= 1 and x <= n-1) and (y >= 1 and y <= n-1):
            if arr[y-1][x-1] != 'X':
                arr[y-1][x-1] += 1 # top left

        if (x >= 0 and x <= n-2) and (y >= 1 and y <= n-1):
            if arr[y-1][x+1] != 'X':
                arr[y-1][x+1] += 1 # top right
        if (x >= 0 and x <= n-1) and (y >= 1 and y <= n-1):
```



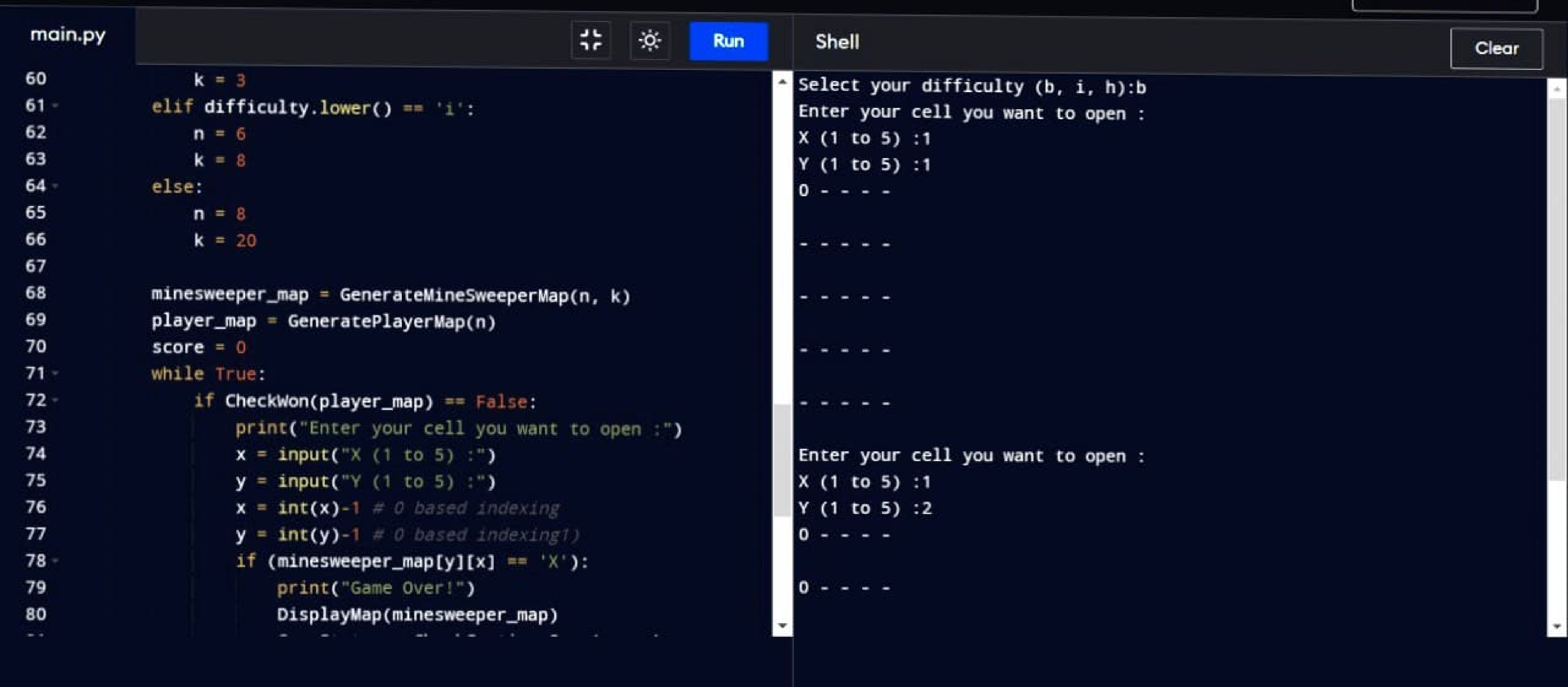
```
            if arr[y-1][x+1] != 'X':
                arr[y-1][x+1] += 1 # top right
        if (x >= 0 and x <= n-1) and (y >= 1 and y <= n-1):
            if arr[y-1][x] != 'X':
                arr[y-1][x] += 1 # top center

        if (x >=0 and x <= n-2) and (y >= 0 and y <= n-2):
            if arr[y+1][x+1] != 'X':
                arr[y+1][x+1] += 1 # bottom right
        if (x >= 1 and x <= n-1) and (y >= 0 and y <= n-2):
            if arr[y+1][x-1] != 'X':
                arr[y+1][x-1] += 1 # bottom left
        if (x >= 0 and x <= n-1) and (y >= 0 and y <= n-2):
            if arr[y+1][x] != 'X':
                arr[y+1][x] += 1 # bottom center
    return arr
def GeneratePlayerMap(n):
    arr = [['-' for row in range(n)] for column in range(n)]
    return arr
def DisplayMap(map):
    for row in map:
        print(" ".join(str(cell) for cell in row))
```



```python
        print(" ".join(str(cell) for cell in row))
        print("")
def CheckWon(map):
    for row in map:
        for cell in row:
            if cell == '-':
                return False
    return True
def CheckContinueGame(score):
    print("Your score: ", score)
    isContinue = input("Do you want to try again? (y/n) :")
    if isContinue == 'n':
        return False
    return True
def Game():
    GameStatus = True
    while GameStatus:
        difficulty = input("Select your difficulty (b, i, h):")
        if difficulty.lower() == 'b':
            n = 5
            k = 3
```

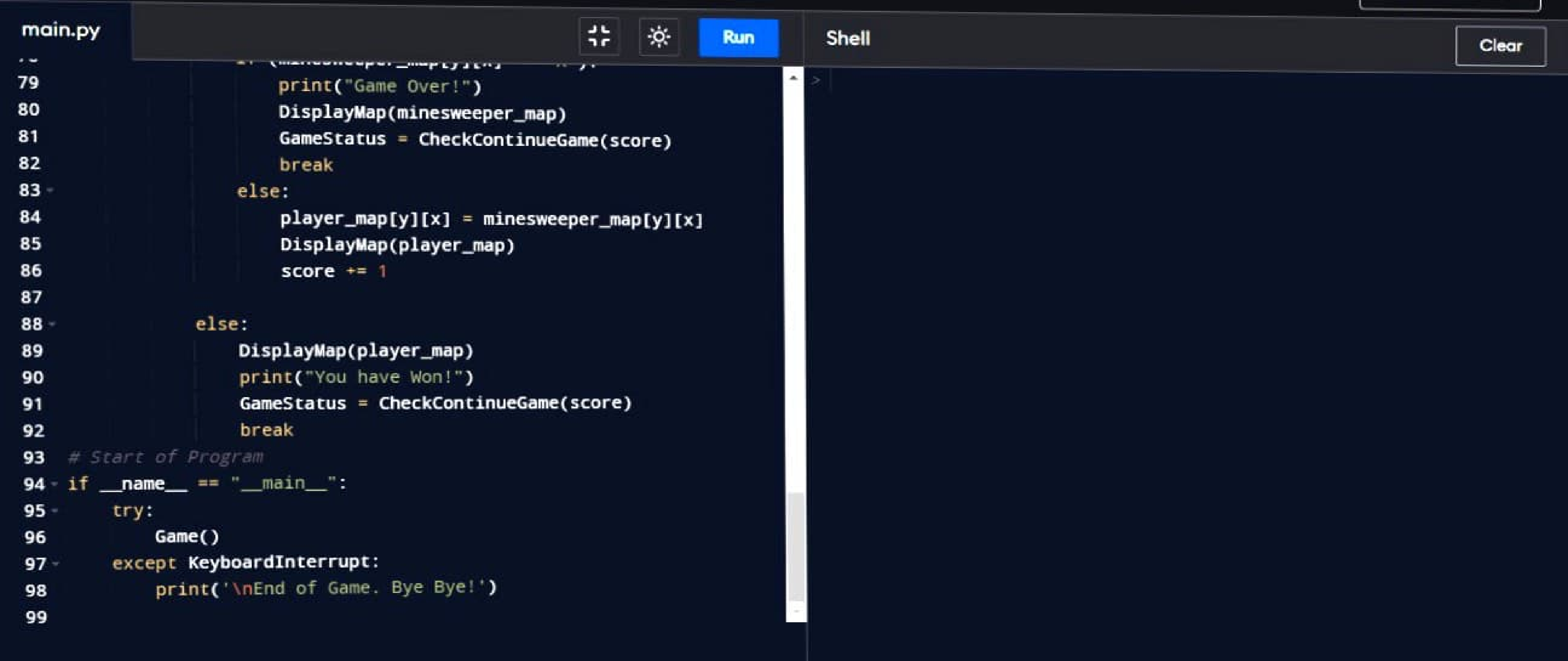


```python
        n = 5
        k = 3
    elif difficulty.lower() == 'i':
        n = 6
        k = 8
    else:
        n = 8
        k = 20

    minesweeper_map = GenerateMineSweeperMap(n, k)
    player_map = GeneratePlayerMap(n)
    score = 0
    while True:
        if CheckWon(player_map) == False:
            print("Enter your cell you want to open :")
            x = input("X (1 to 5) :")
            y = input("Y (1 to 5) :")
            x = int(x)-1 # 0 based indexing
            y = int(y)-1 # 0 based indexing1)
            if (minesweeper_map[y][x] == 'X'):
                print("Game Over!")
                DisplayMap(minesweeper_map)
```

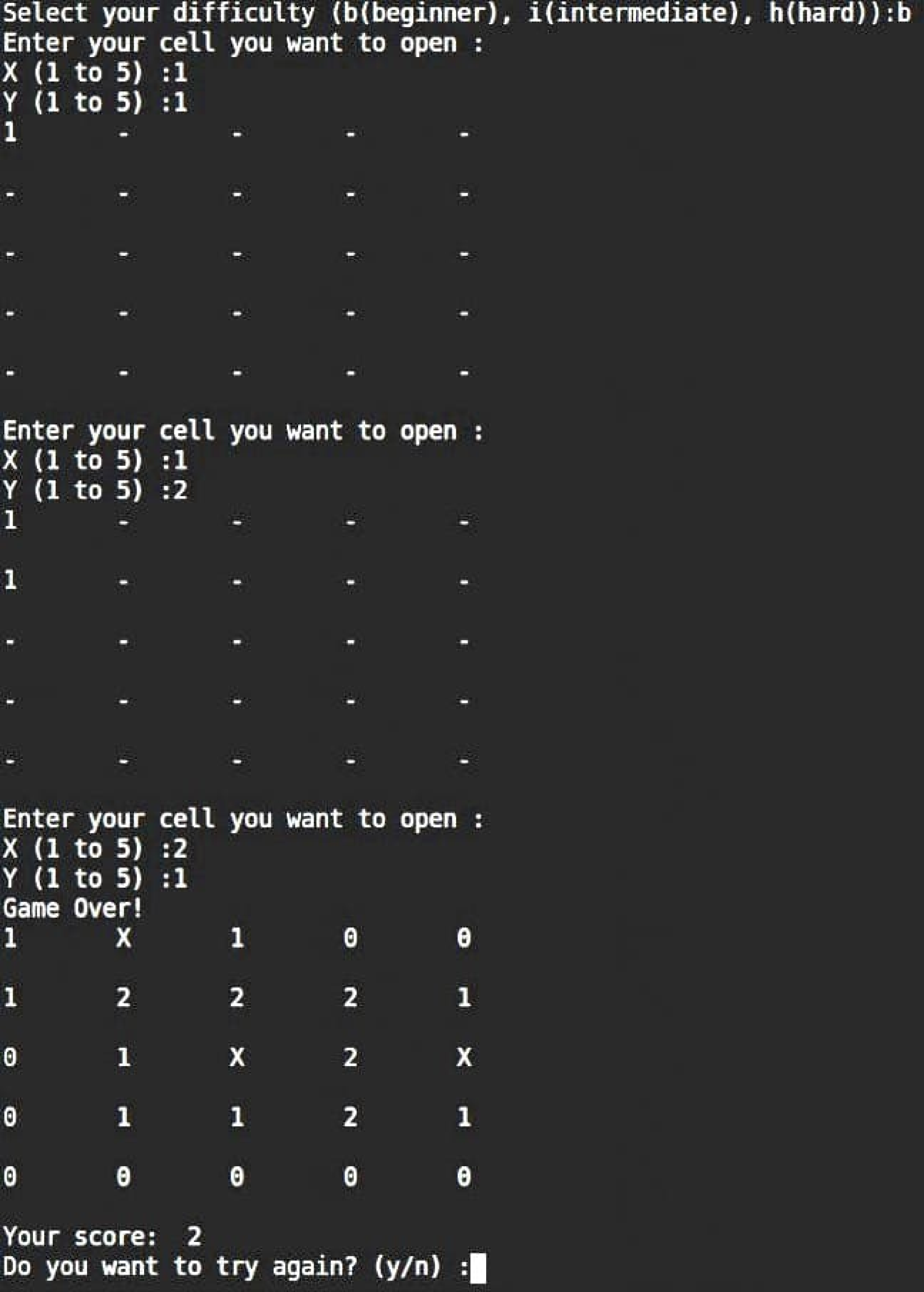


```python
        k = 3
    elif difficulty.lower() == 'i':
        n = 6
        k = 8
    else:
        n = 8
        k = 20

    minesweeper_map = GenerateMineSweeperMap(n, k)
    player_map = GeneratePlayerMap(n)
    score = 0
    while True:
        if CheckWon(player_map) == False:
            print("Enter your cell you want to open :")
            x = input("X (1 to 5) :")
            y = input("Y (1 to 5) :")
            x = int(x)-1 # 0 based indexing
            y = int(y)-1 # 0 based indexing1)
            if (minesweeper_map[y][x] == 'X'):
                print("Game Over!")
                DisplayMap(minesweeper_map)
```

```
Select your difficulty (b, i, h):b
Enter your cell you want to open :
X (1 to 5) :1
Y (1 to 5) :1
0 - - - -

- - - - -

- - - - -

- - - - -

- - - - -

Enter your cell you want to open :
X (1 to 5) :1
Y (1 to 5) :2
0 - - - -

0 - - - -
```



```python
                if (minesweeper_map[y][x] == "X"):
                    print("Game Over!")
                    DisplayMap(minesweeper_map)
                    GameStatus = CheckContinueGame(score)
                    break
                else:
                    player_map[y][x] = minesweeper_map[y][x]
                    DisplayMap(player_map)
                    score += 1

            else:
                DisplayMap(player_map)
                print("You have Won!")
                GameStatus = CheckContinueGame(score)
                break
# Start of Program
if __name__ == "__main__":
    try:
        Game()
    except KeyboardInterrupt:
        print('\nEnd of Game. Bye Bye!')

```



```
Select your difficulty (b(beginner), i(intermediate), h(hard)):b
Enter your cell you want to open :
X (1 to 5) :1
Y (1 to 5) :1
1       -       -       -       -

-       -       -       -       -

-       -       -       -       -

-       -       -       -       -

-       -       -       -       -

Enter your cell you want to open :
X (1 to 5) :1
Y (1 to 5) :2
1       -       -       -       -

1       -       -       -       -

-       -       -       -       -

-       -       -       -       -

-       -       -       -       -

Enter your cell you want to open :
X (1 to 5) :2
Y (1 to 5) :1
Game Over!
1       X       1       0       0

1       2       2       2       1

0       1       X       2       X

0       1       1       2       1

0       0       0       0       0

Your score:  2
Do you want to try again? (y/n) :
```

Bonafide Certificate

Certified that this project "Minesweeper" is the bonafide work of ,

P. Sahith arya Charan

B. Mohan Sai

E. Vishnu Vardhan Reddy

who carried out the project work under my Supervision.

Dr. Dhanpratap Singh
25706
Department of
Intelligent systems.