S. B. JAIN INSTITUTE OF TECHNOLOGY, MANAGEMENT & RESEARCH, NAGPUR.

Practical No. 0

Aim: PRELAB – Write a Python Program to implement magic square.

Name of Student: Shrutika Pradeep Bagdi

Roll No.: CS22130

Semester/Year: V/III

Academic Session: 2024-2025

Date of Performance:

Date of Submission:

Aim: PRELAB – Write a Python program to implement magic square.

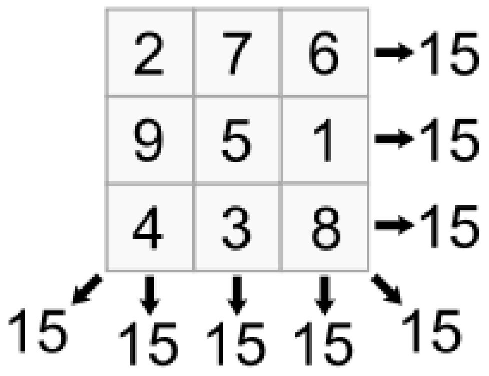
Objective/Expected Learning Outcome:

- To be able to understand the concept of Magic Square.
- To be able to acquire the puzzle solving capability.

Theory:

A magic square of order n is an arrangement of n^2 numbers, usually distinct integers, in a square, such that the n numbers in all rows, all columns, and both diagonals sum to the same constant. A magic square contains the integers from 1 to n^2.
The constant sum in every row, column and diagonal are called the magic constant or magic sum, M. The magic constant of a normal magic square depends only on n and has the following value:
$M = n(n^2+1)/2$
For normal magic squares of order n = 3, 4, 5, ...,

The magic constants are: 15, 34, 65, 111, 175, 260, ...



In any magic square, the first number i.e. 1 is stored at position (n/2, n-1). Let this position be (i,j). The next number is stored at position (i-1, j+1) where we can consider each row & column as circular array i.e. they wrap around.
Three conditions hold:

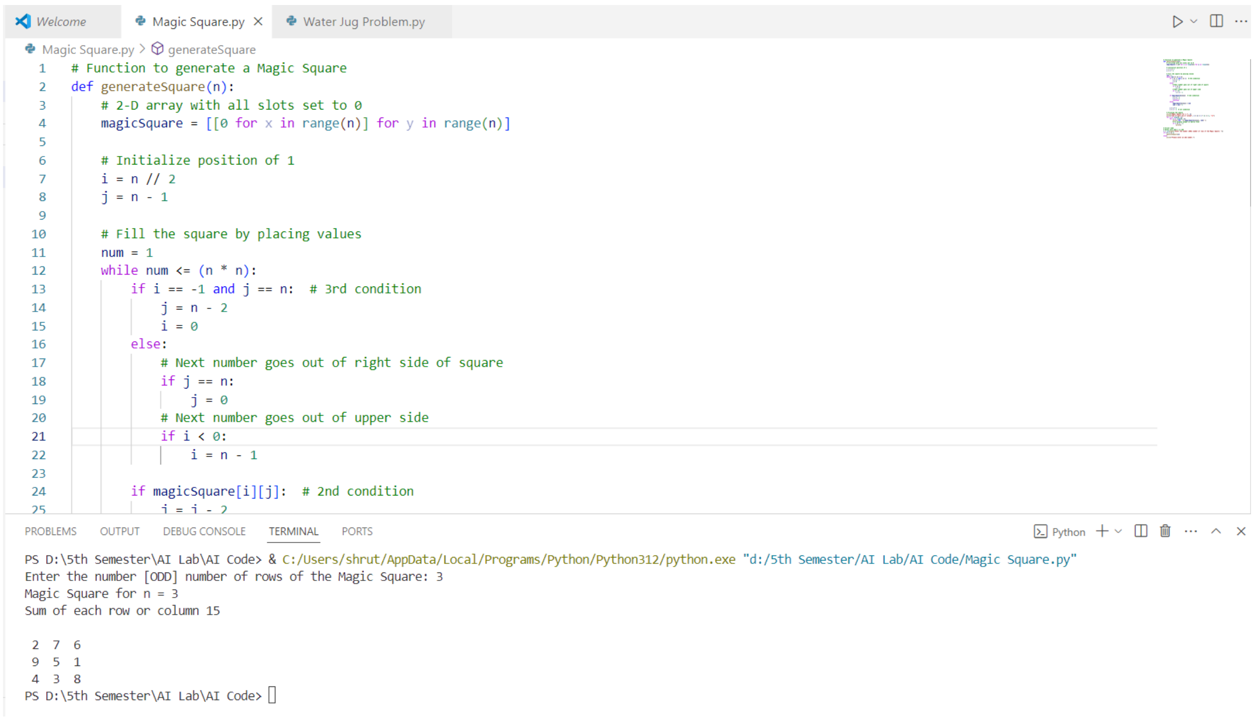
1. The position of next number is calculated by decrementing row number of the previous number by 1, and incrementing the column number of the previous number by 1. At any time, if the calculated row position becomes -1, it will wrap around to n-1. Similarly, if the calculated column position becomes n, it will wrap around to 0.
2. If the magic square already contains a number at the calculated position, calculated column position will be decremented by 2, and calculated row position will be incremented by 1.

3. If the calculated row position is -1 & calculated column position is n, the new position would be: (0, n-2).

Program Code:

```
# Function to generate a Magic Square
def generateSquare(n):
    # 2-D array with all slots set to 0
    magicSquare = [[0 for x in range(n)] for y in range(n)]

    # Initialize position of 1
    i = n // 2
    j = n - 1

    # Fill the square by placing values
    num = 1
    while num <= (n * n):
        if i == -1 and j == n:  # 3rd condition
            j = n - 2
            i = 0
        else:
            # Next number goes out of right side of square
            if j == n:
                j = 0
            # Next number goes out of upper side
            if i < 0:
                i = n - 1

        if magicSquare[i][j]:  # 2nd condition
            j = j - 2
            i = i + 1
            continue
        else:
            magicSquare[i][j] = num
            num = num + 1

        j = j + 1
        i = i - 1  # 1st condition

    # Printing the square
    print("Magic Square for n =", n)
    print("Sum of each row or column", n * (n * n + 1) // 2, "\n")
    for i in range(0, n):
        for j in range(0, n):
            print('%2d ' % (magicSquare[i][j]), end='')
            # To display output in matrix form
            if j == n - 1:
                print()

# Driver Code
# Works only when n is odd
n = int(input("Enter the number [ODD] number of rows of the Magic Square: "))
if n % 2 == 1:
    generateSquare(n)
else:
    print("Please enter an odd number.")
```

Output:



```python
# Function to generate a Magic Square
def generateSquare(n):
    # 2-D array with all slots set to 0
    magicSquare = [[0 for x in range(n)] for y in range(n)]

    # Initialize position of 1
    i = n // 2
    j = n - 1

    # Fill the square by placing values
    num = 1
    while num <= (n * n):
        if i == -1 and j == n:  # 3rd condition
            j = n - 2
            i = 0
        else:
            # Next number goes out of right side of square
            if j == n:
                j = 0
            # Next number goes out of upper side
            if i < 0:
                i = n - 1

        if magicSquare[i][j]:  # 2nd condition
            i = i - 2
```

```
PS D:\5th Semester\AI Lab\AI Code> & C:/Users/shrut/AppData/Local/Programs/Python/Python312/python.exe "d:/5th Semester/AI Lab/AI Code/Magic Square.py"
Enter the number [ODD] number of rows of the Magic Square: 3
Magic Square for n = 3
Sum of each row or column 15

 2  7  6
 9  5  1
 4  3  8
PS D:\5th Semester\AI Lab\AI Code> 
```

Conclusion:

Thus I created a magic squar box and got to know that it only works for odd no. of rows and columns.

Discussion Questions:

Q.1) What is Magic Square?

Q.2) Complete the given magic square whose sum of each row, column and diagonal is 72.

Q.3) What is the optimal way of generating all possible 3x3 magic squares?

Q.4) How many possible 3x3 magic squares are there?

Q.5) What is Ramanujan magic square?

References:

https://byjus.com/maths/magic-square/

https://www.geeksforgeeks.org/magic-square/

https://mathworld.wolfram.com/MagicSquare.html