**DAY 24.06.24 LAB PROGRAMS**

**1.PERMUTATIONS**

def permutations(nums):

if not nums:

return [[]]

dp=[[]for \_ in range(len(nums)+1)]

dp[0]=[[]]

for i in range(1,len(nums)+1):

current\_num=nums[i-1]

for perm in dp[i-1]:

for j in range(i):

new=perm[:]

new.insert(j,current\_num)

dp[i].append(new)

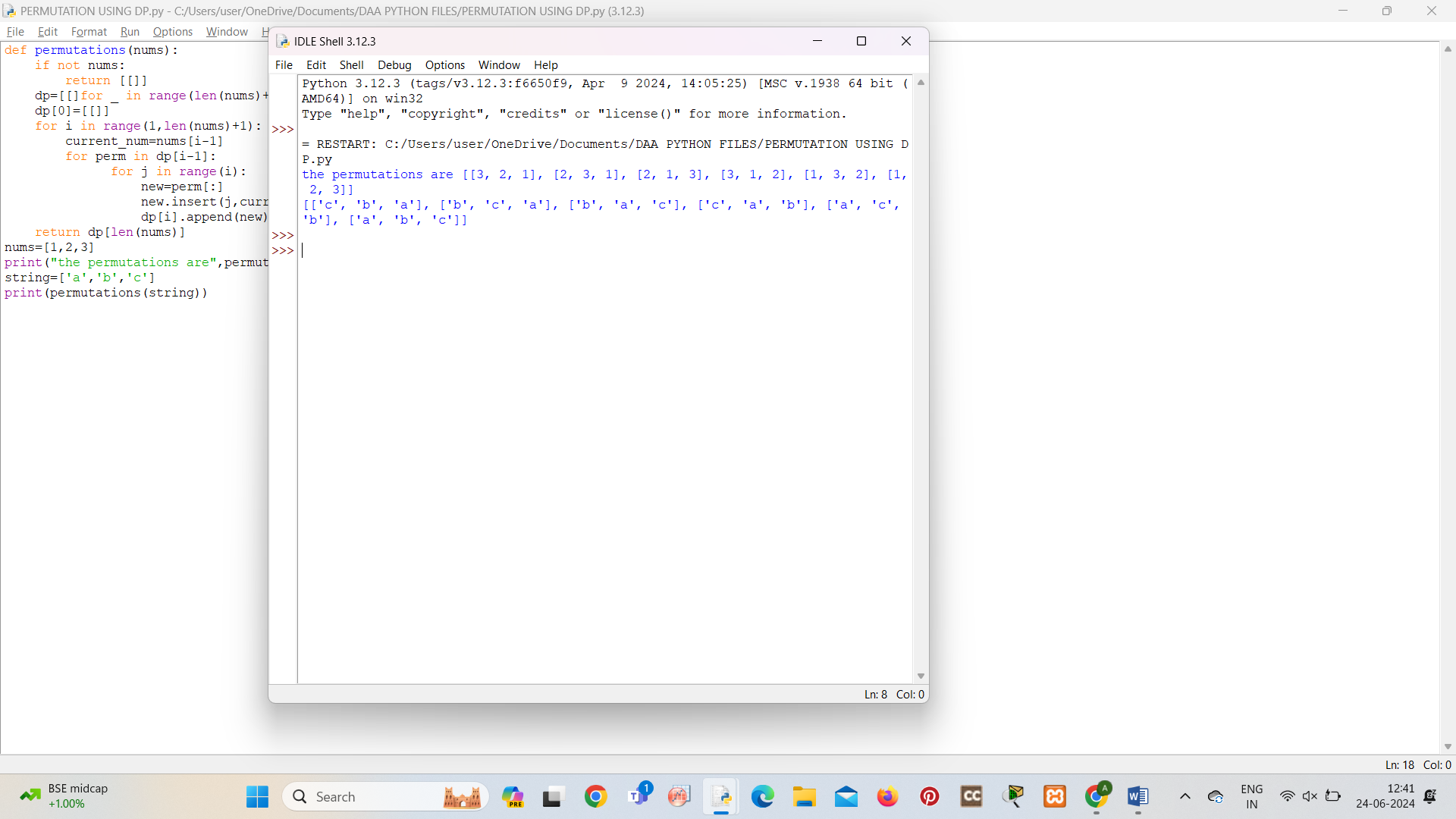
return dp[len(nums)]

nums=[1,2,3]

print("the permutations are",permutations(nums))

string=['a','b','c']

print(permutations(string))



**2.COMBINATIONS**

def combinations(nums,k):

n=len(nums)

if k>n:

return [[]]

dp=[[]for \_ in range(k+1)]

dp[0]=[[]]

for i in nums:

for j in range(min(k,len(dp)),0,-1):

for comb in dp[j-1]:

new=comb[:]

new.append(i)

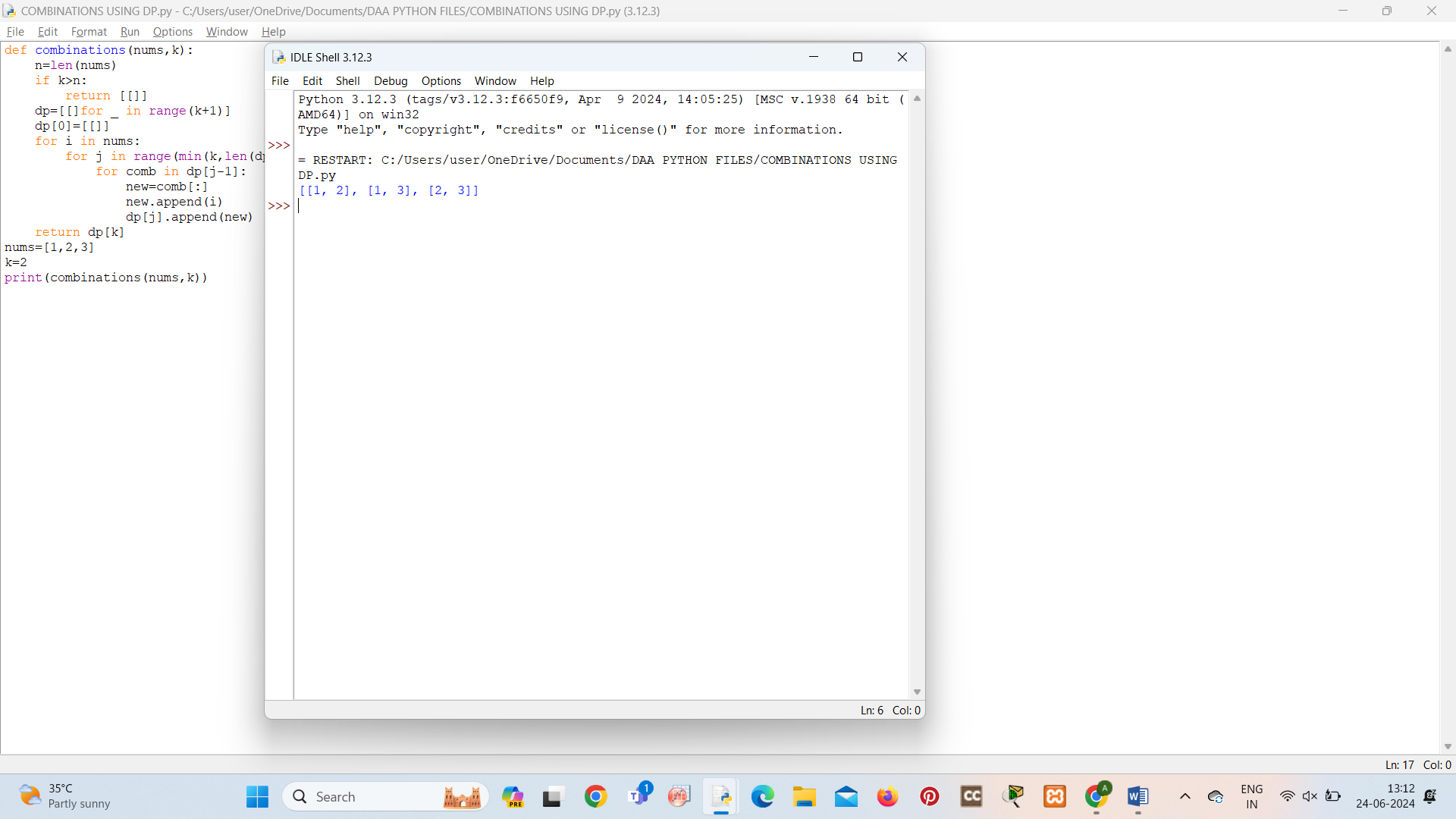
dp[j].append(new)

return dp[k]

nums=[1,2,3]

k=2

print(combinations(nums,k))



**3.SUBSET GENERATION**

def subset(nums):

subsets=[]

def backtrack(start,current\_subset):

subsets.append(current\_subset[:])

for i in range(start,len(nums)):

current\_subset.append(nums[i])

backtrack(i+1,current\_subset)

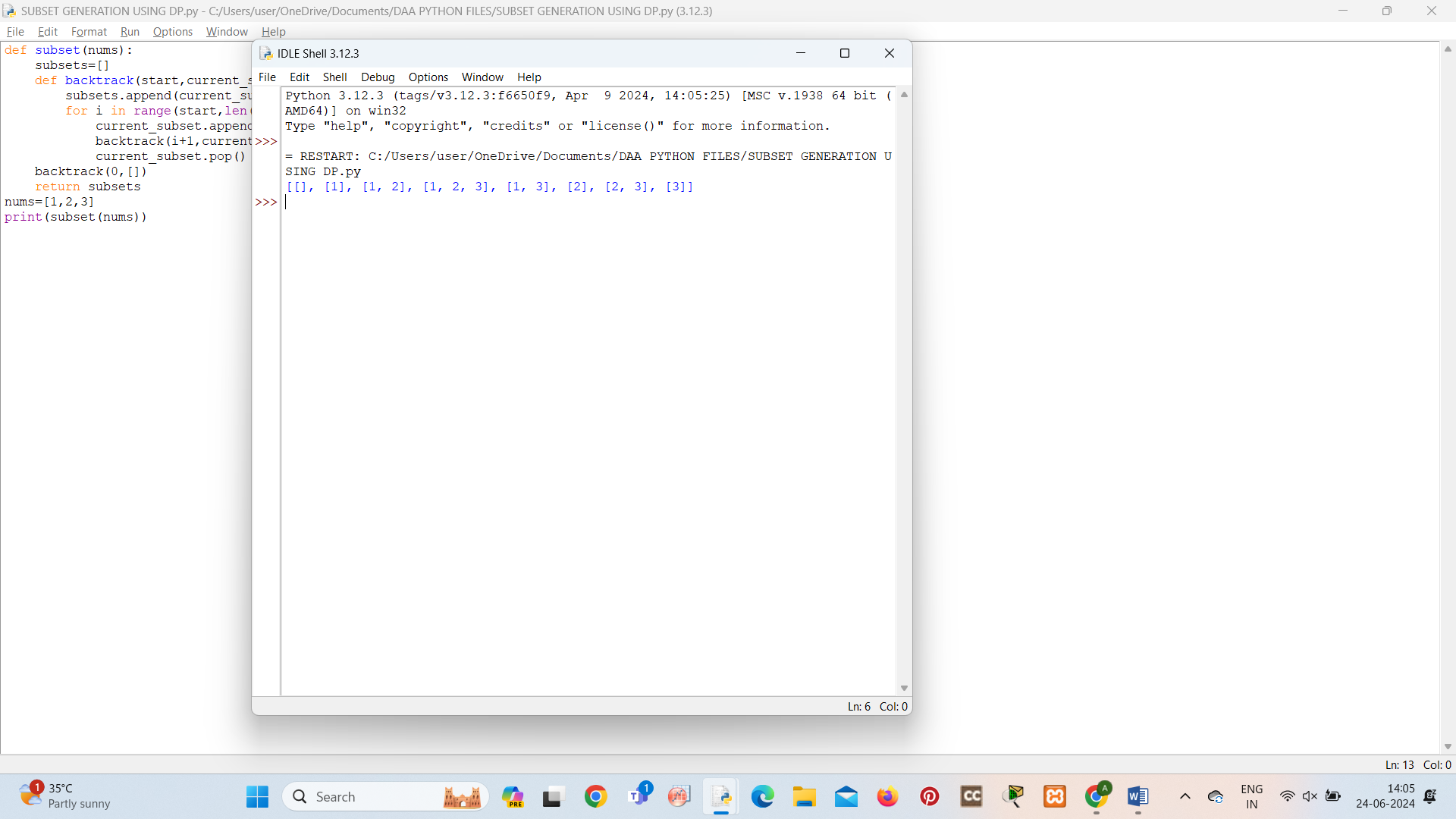
current\_subset.pop()

backtrack(0,[])

return subsets

nums=[1,2,3]

print(subset(nums))



**4.HAMILTONIAN CIRCUIT**

def hamiltonian\_circuit(graph):

n = len(graph)

start = 0

dp = [[False] \* n for \_ in range(1 << n)]

dp[1 << start][start] = True

for mask in range(1 << n):

for i in range(n):

if dp[mask][i]:

for j in range(n):

if graph[i][j] and j != i and not (mask & (1 << j)):

dp[mask | (1 << j)][j] = True

return any(dp[(1 << n) - 1][j] and graph[j][start] for j in range(n))

graph = [

[0, 1, 1, 0],

[1, 0, 1, 1],

[1, 1, 0, 1],

[0, 1, 1, 0]

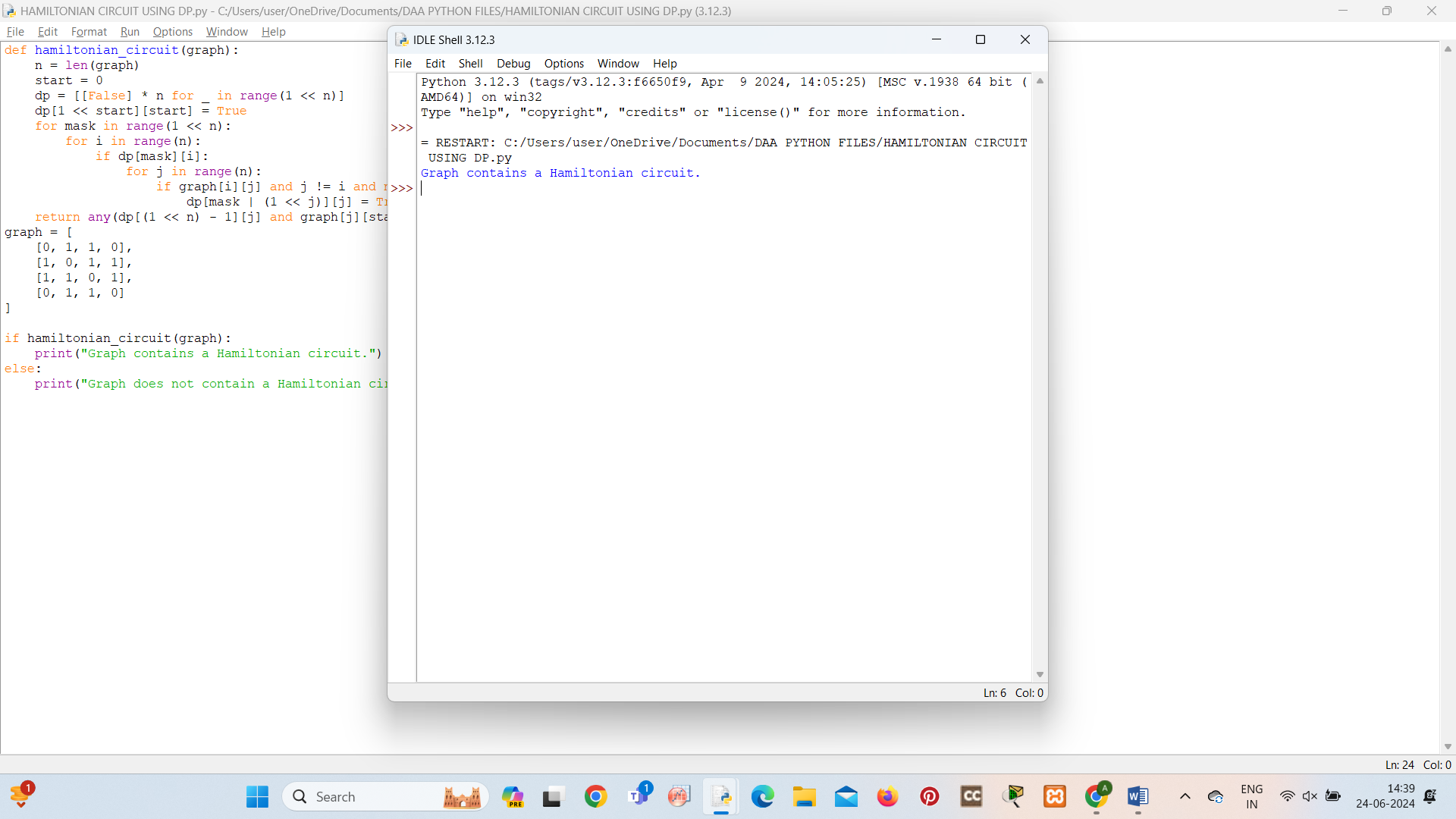
]

if hamiltonian\_circuit(graph):

print("Graph contains a Hamiltonian circuit.")

else:

print("Graph does not contain a Hamiltonian circuit.")



**5.SUDOKU SOLVER**

def solve\_sudoku(board):

def is\_valid(num, row, col):

for i in range(9):

if board[row][i] == num or board[i][col] == num or board[(row//3)\*3 + i//3][(col//3)\*3 + i%3] == num:

return False

return True

def solve():

for i in range(9):

for j in range(9):

if board[i][j] == 0:

for num in range(1, 10):

if is\_valid(num, i, j):

board[i][j] = num

if solve():

return True

board[i][j] = 0

return False

return True

solve()

board = [

[5, 3, 0, 0, 7, 0, 0, 0, 0],

[6, 0, 0, 1, 9, 5, 0, 0, 0],

[0, 9, 8, 0, 0, 0, 0, 6, 0],

[8, 0, 0, 0, 6, 0, 0, 0, 3],

[4, 0, 0, 8, 0, 3, 0, 0, 1],

[7, 0, 0, 0, 2, 0, 0, 0, 6],

[0, 6, 0, 0, 0, 0, 2, 8, 0],

[0, 0, 0, 4, 1, 9, 0, 0, 5],

[0, 0, 0, 0, 8, 0, 0, 7, 9]

]

solve\_sudoku(board)

for row in board:

print(row)

