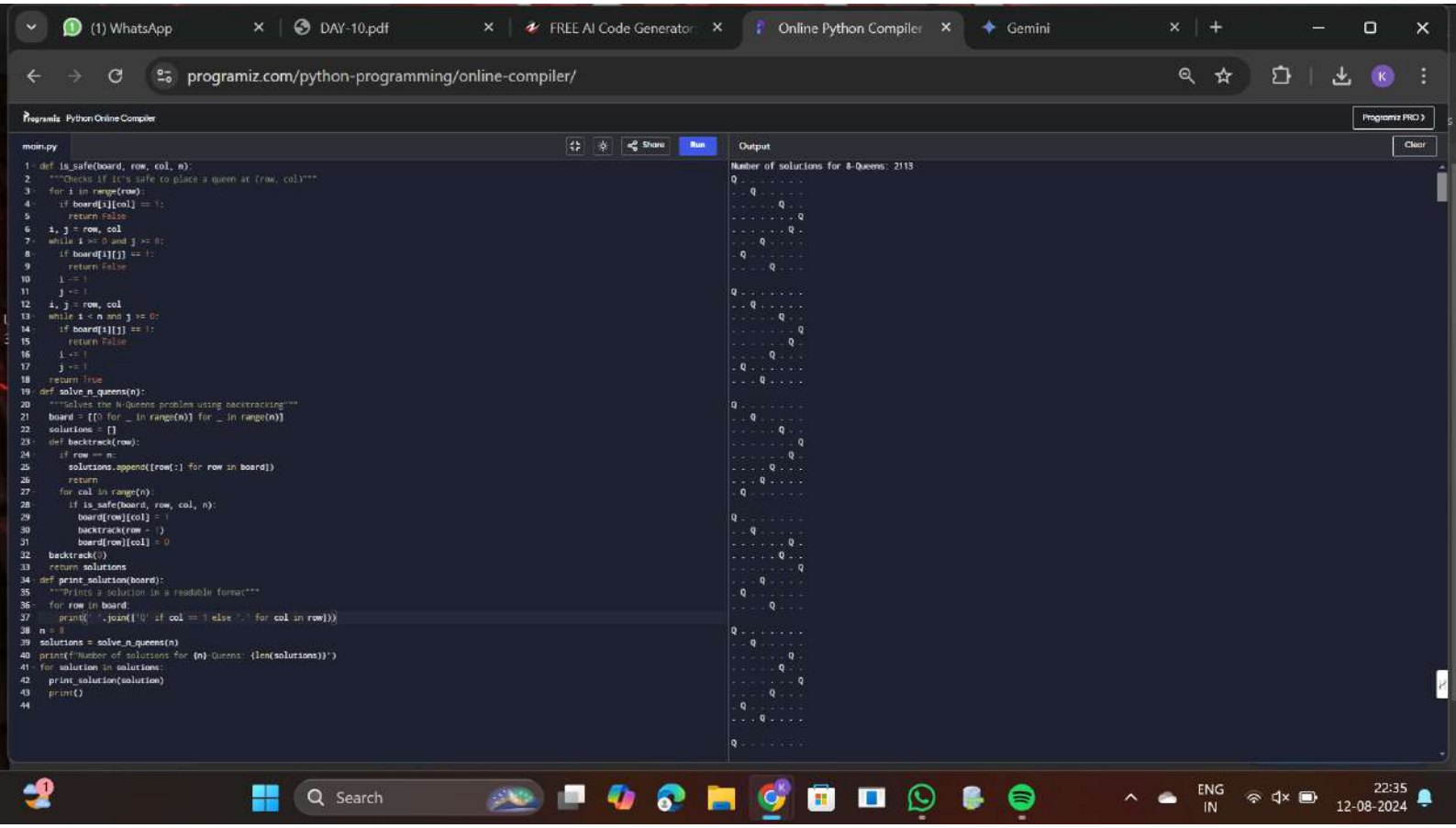
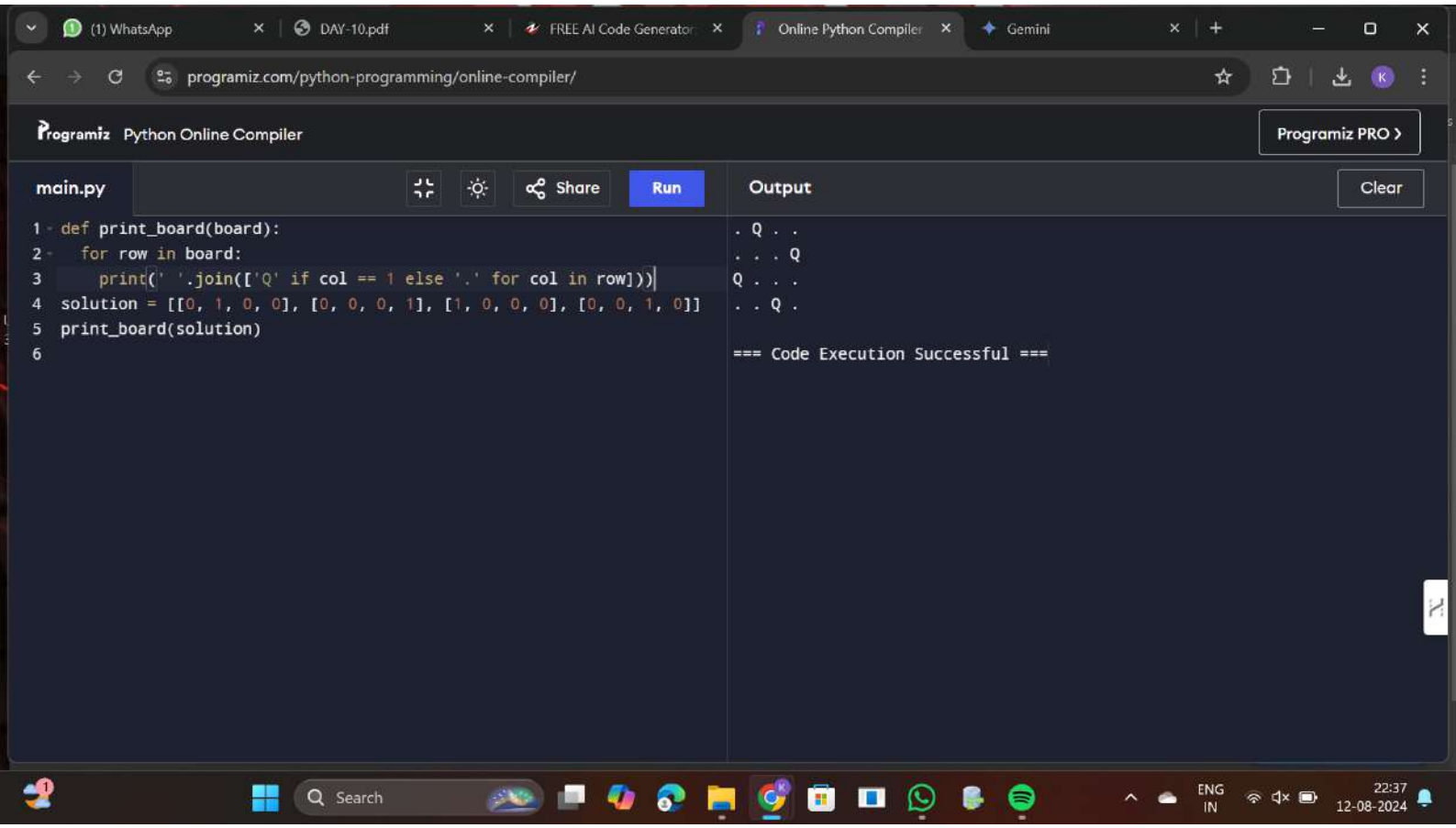
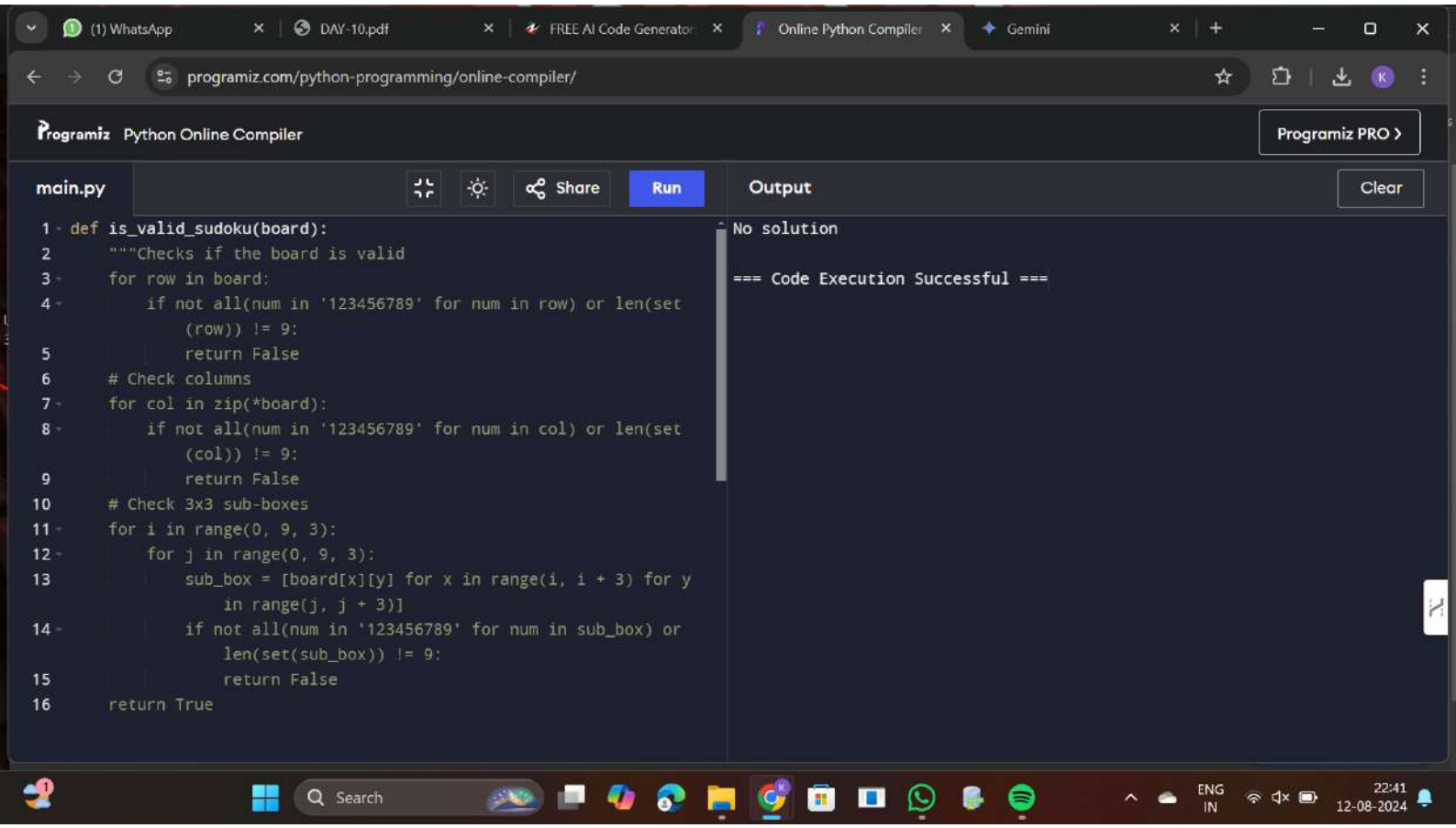
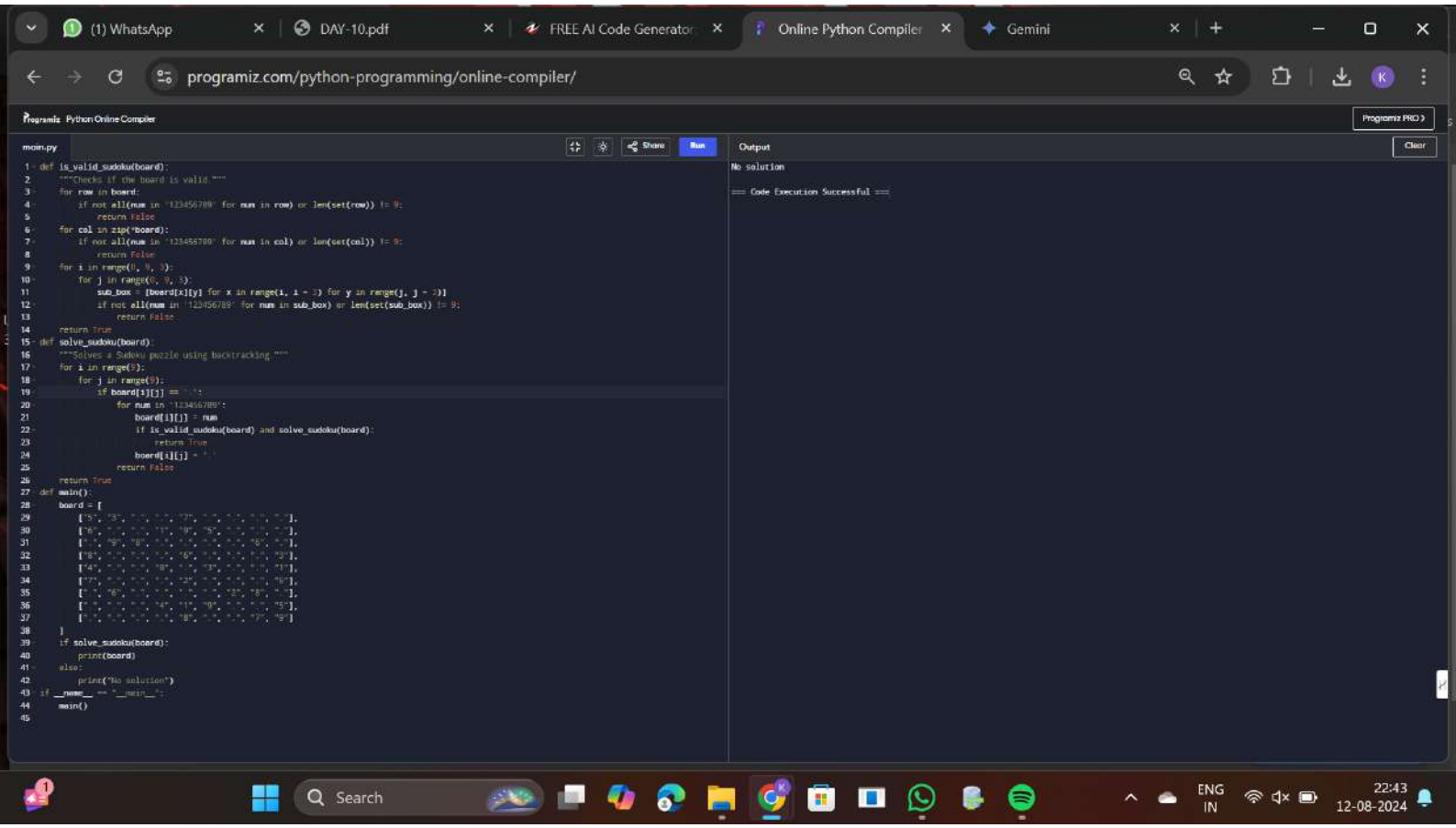
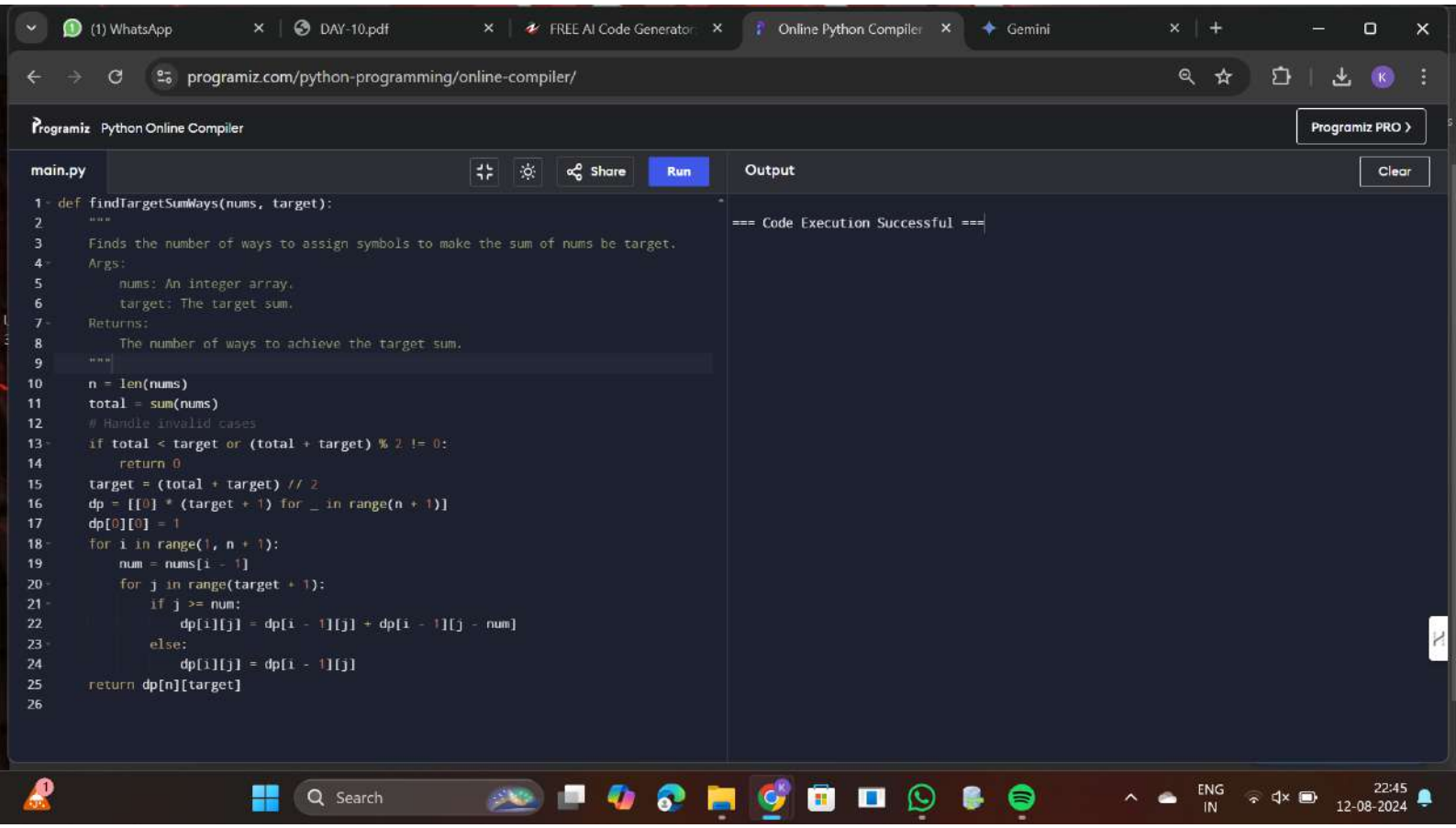
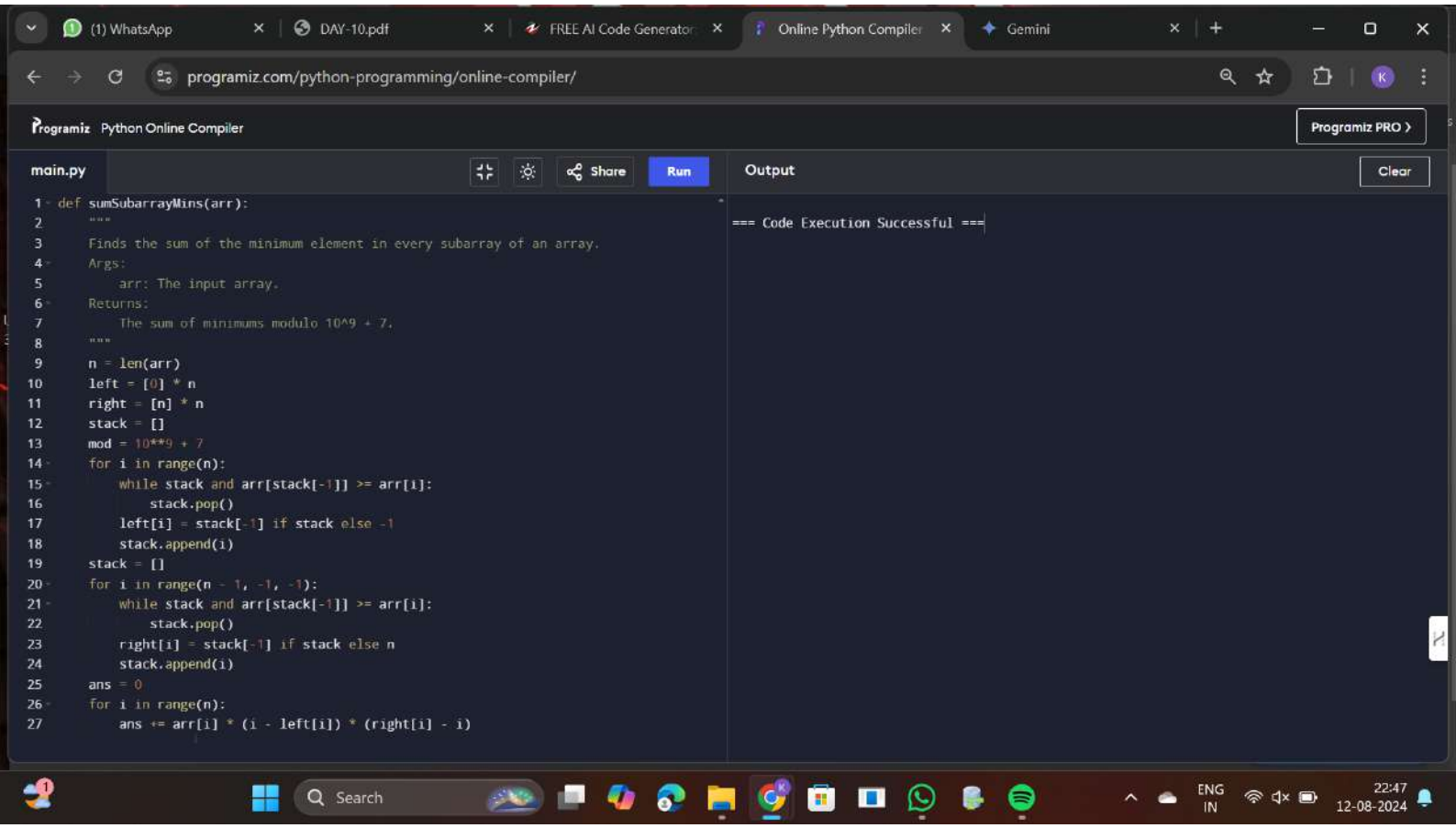

(1) WhatsApp
DAY-10.pdf
FREE AI Code Generator
Online Python Compiler
Gemini
programiz.com/python-programming/online-compiler/
Programiz Python Online Compiler
Programiz PRO
main.py
Share
Run
Output
Clear
def is_safe(board, row, col, n):
"""Checks if it's safe to place a queen at (row, col)"""
for i in range(row):
if board[i][col] == 1:
return False
i, j = row, col
while i >= 0 and j >= 0:
if board[i][j] == 1:
return False
i -= 1
j -= 1
i, j = row, col
while i < n and j >= 0:
if board[i][j] == 1:
return False
i += 1
j -= 1
return True
def solve_n_queens(n):
"""Solves the N-Queens problem using backtracking"""
board = [[0 for _ in range(n)] for _ in range(n)]
solutions = []
def backtrack(row):
if row == n:
solutions.append([row[:] for row in board])
return
for col in range(n):
if is_safe(board, row, col, n):
board[row][col] = 1
backtrack(row + 1)
board[row][col] = 0
backtrack(0)
return solutions
def print_solution(board):
"""Prints a solution in a readable format"""
for row in board:
print(" ".join(['Q' if col == 1 else '.' for col in row]))
n = 8
solutions = solve_n_queens(n)
print(f"Number of solutions for {n}-Queens: {len(solutions)}")
for solution in solutions:
print_solution(solution)
print()
Number of solutions for 8-Queens: 2113
Search
ENG
IN
22:35
12-08-2024

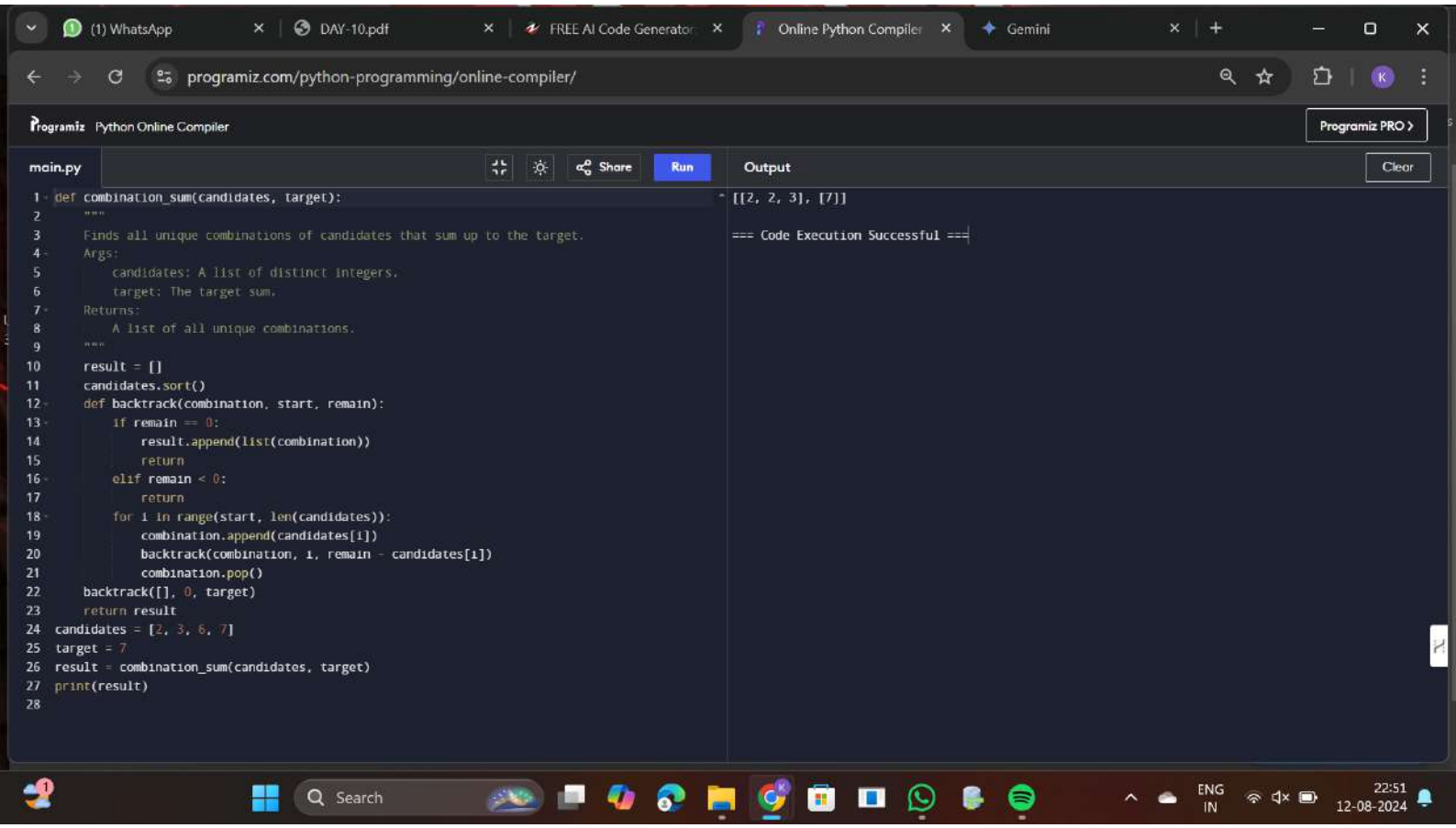

(1) WhatsApp
DAY-10.pdf
FREE AI Code Generator
Online Python Compiler
Gemini
programiz.com/python-programming/online-compiler/
Programiz Python Online Compiler
Programiz PRO
main.py
Share
Run
Output
Clear
def print_board(board):
for row in board:
print(' '.join(['Q' if col == 1 else '.' for col in row]))
solution = [[0, 1, 0, 0], [0, 0, 0, 1], [1, 0, 0, 0], [0, 0, 1, 0]]
print_board(solution)
. Q . .
. . . Q
Q . . .
. . Q .
=== Code Execution Successful ===
Search
ENG
IN
22:37
12-08-2024

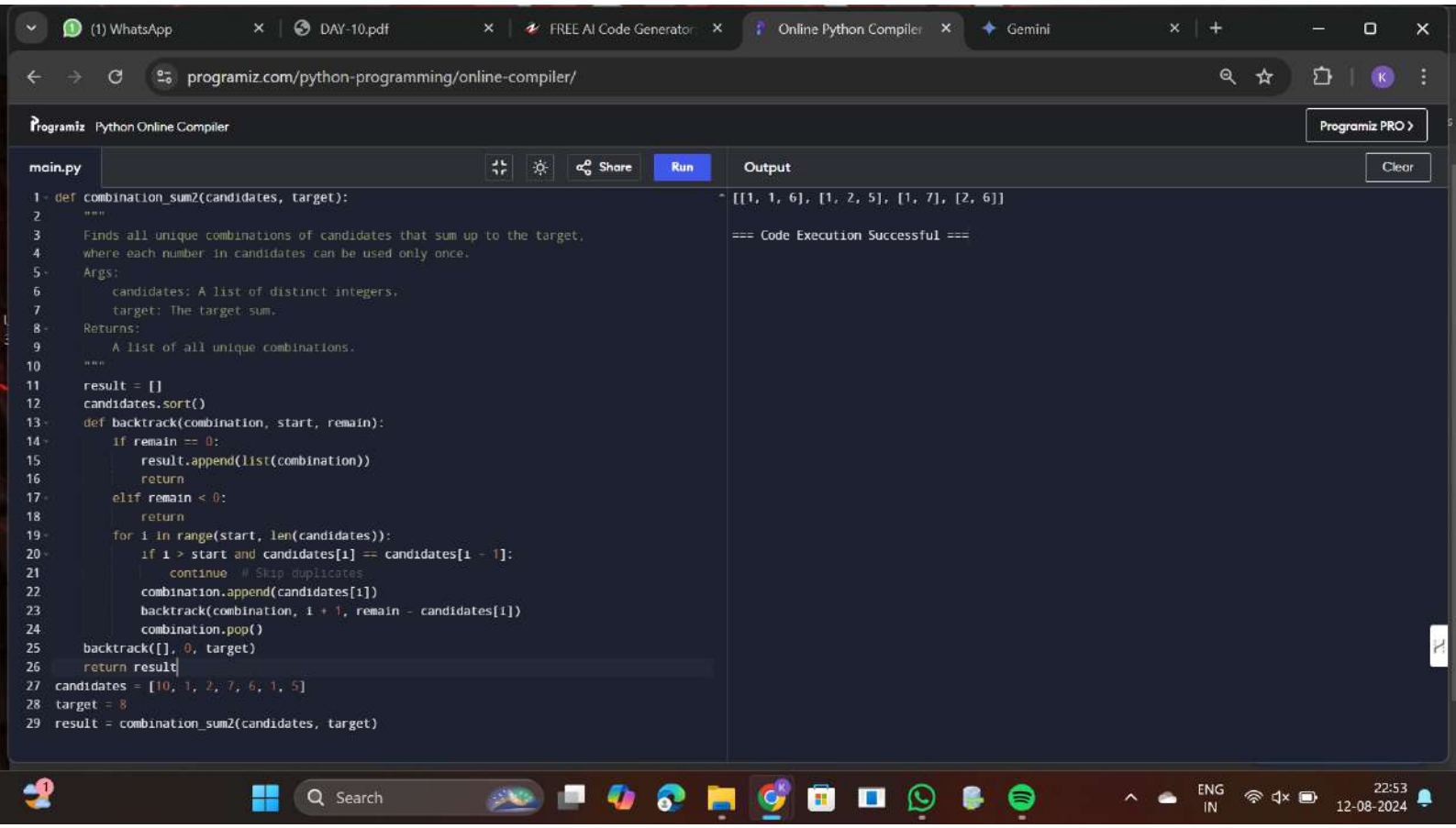

(1) WhatsApp
DAY-10.pdf
FREE AI Code Generator
Online Python Compiler
Gemini
programiz.com/python-programming/online-compiler/
Programiz Python Online Compiler
Programiz PRO
main.py
Share
Run
Output
Clear
def is_valid_sudoku(board):
"""Checks if the board is valid
for row in board:
if not all(num in '123456789' for num in row) or len(set(row)) != 9:
return False
# Check columns
for col in zip(*board):
if not all(num in '123456789' for num in col) or len(set(col)) != 9:
return False
# Check 3x3 sub-boxes
for i in range(0, 9, 3):
for j in range(0, 9, 3):
sub_box = [board[x][y] for x in range(i, i + 3) for y in range(j, j + 3)]
if not all(num in '123456789' for num in sub_box) or len(set(sub_box)) != 9:
return False
return True
No solution
=== Code Execution Successful ===
Search
ENG
IN
22:41
12-08-2024

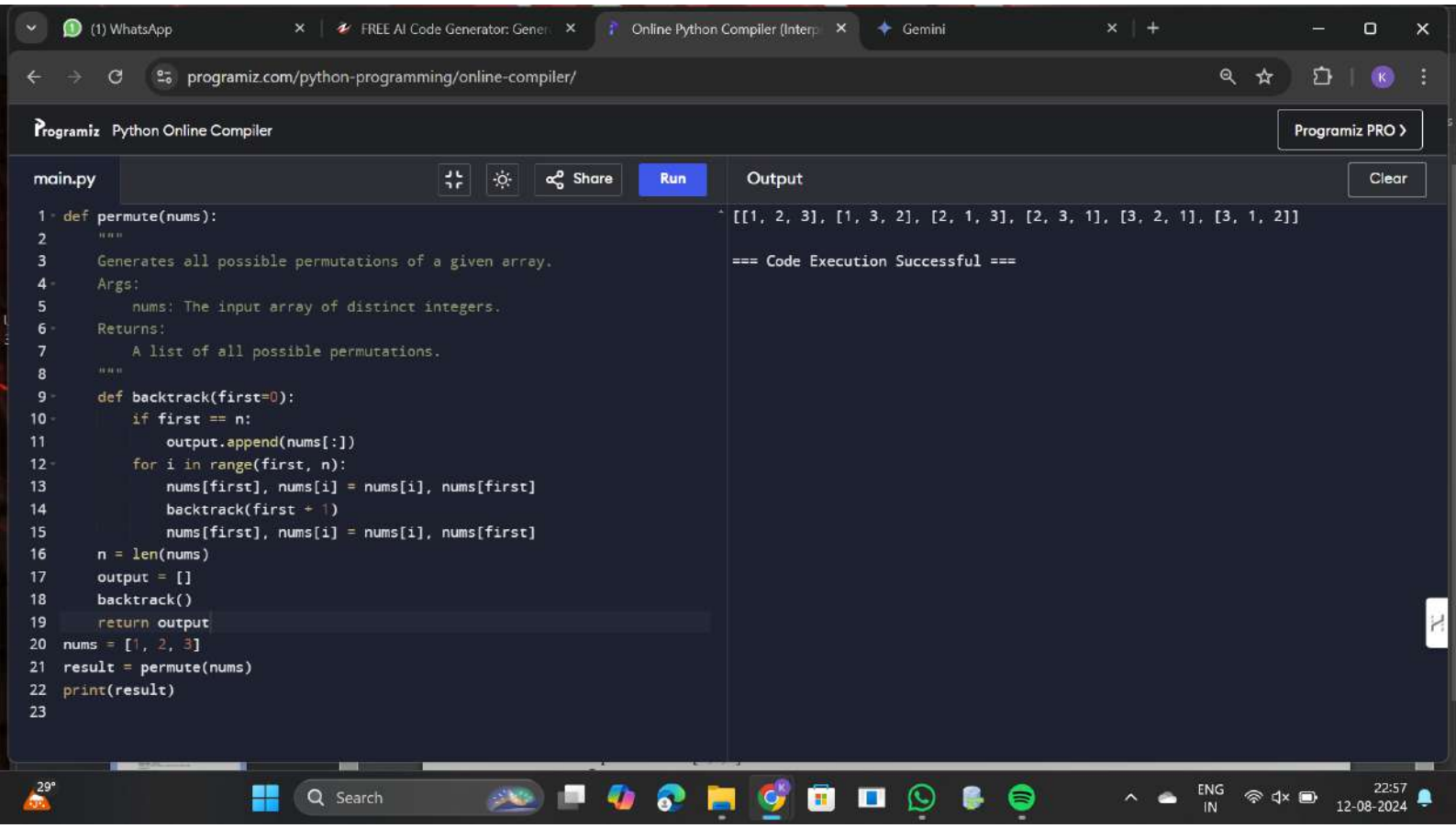

(1) WhatsApp
DAY-10.pdf
FREE AI Code Generator
Online Python Compiler
Gemini
programiz.com/python-programming/online-compiler/
Programiz Python Online Compiler
Programiz PRO >
main.py
Share
Run
Output
Clear
No solution
== Code Execution Successful ==

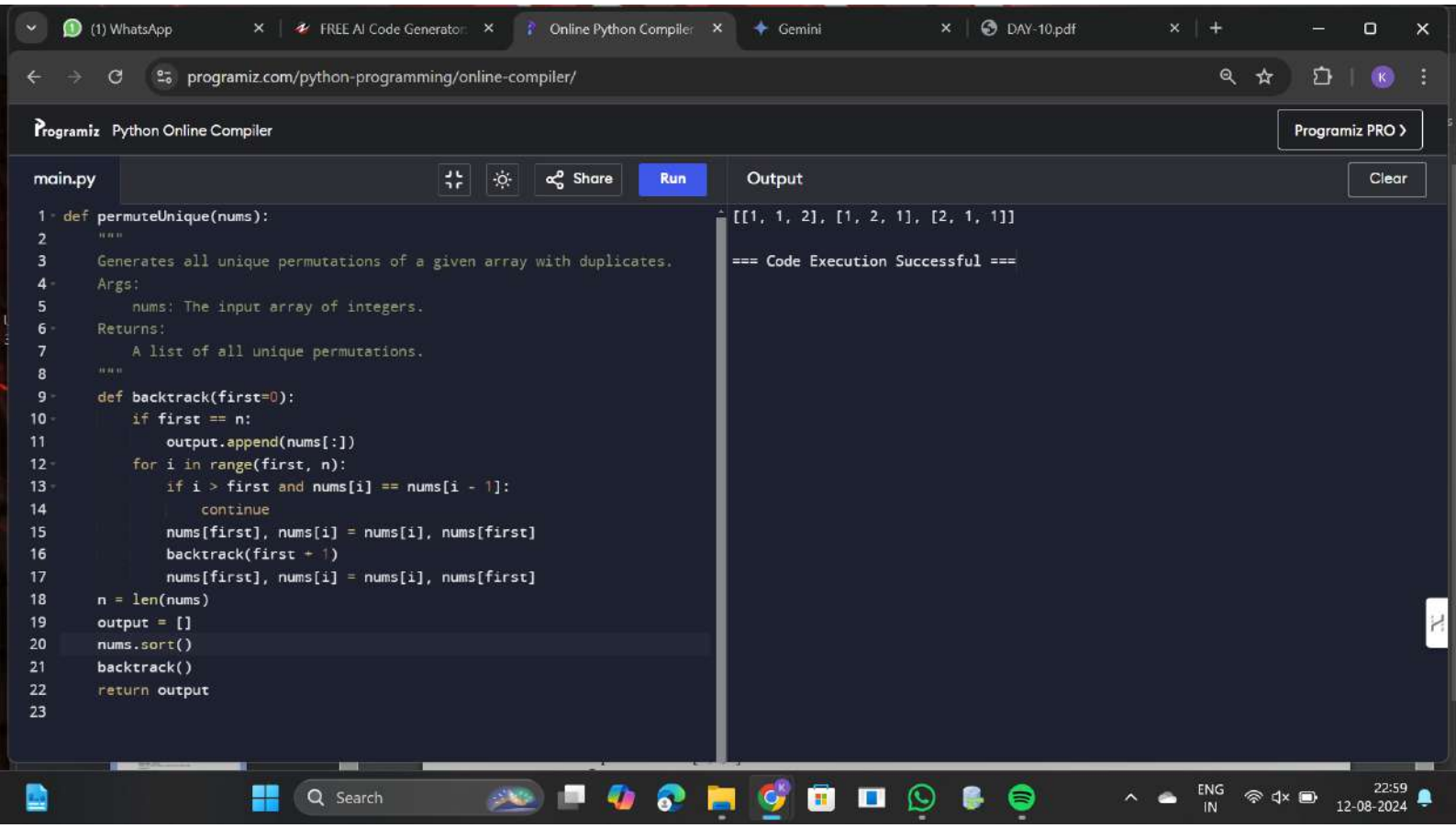

(1) WhatsApp
DAY-10.pdf
FREE AI Code Generator
Online Python Compiler
Gemini
programiz.com/python-programming/online-compiler/
Programiz Python Online Compiler
Programiz PRO
main.py
Share
Run
Output
Clear
def findTargetSumWays(nums, target):
"""
Finds the number of ways to assign symbols to make the sum of nums be target.
Args:
nums: An integer array.
target: The target sum.
Returns:
The number of ways to achieve the target sum.
"""
n = len(nums)
total = sum(nums)
# Handle invalid cases
if total < target or (total + target) % 2 != 0:
return 0
target = (total + target) // 2
dp = [[0] * (target + 1) for _ in range(n + 1)]
dp[0][0] = 1
for i in range(1, n + 1):
num = nums[i - 1]
for j in range(target + 1):
if j >= num:
dp[i][j] = dp[i - 1][j] + dp[i - 1][j - num]
else:
dp[i][j] = dp[i - 1][j]
return dp[n][target]
=== Code Execution Successful ===
Search
ENG
IN
22:45
12-08-2024


(1) WhatsApp
DAY-10.pdf
FREE AI Code Generator
Online Python Compiler
Gemini
programiz.com/python-programming/online-compiler/
Programiz Python Online Compiler
Programiz PRO
main.py
Share
Run
Output
Clear
def sumSubarrayMins(arr):
"""
Finds the sum of the minimum element in every subarray of an array.
Args:
arr: The input array.
Returns:
The sum of minimums modulo 10^9 + 7.
"""
n = len(arr)
left = [0] * n
right = [n] * n
stack = []
mod = 10**9 + 7
for i in range(n):
while stack and arr[stack[-1]] >= arr[i]:
stack.pop()
left[i] = stack[-1] if stack else -1
stack.append(i)
stack = []
for i in range(n - 1, -1, -1):
while stack and arr[stack[-1]] >= arr[i]:
stack.pop()
right[i] = stack[-1] if stack else n
stack.append(i)
ans = 0
for i in range(n):
ans += arr[i] * (i - left[i]) * (right[i] - i)
=== Code Execution Successful ===
Search
ENG
IN
22:47
12-08-2024


(1) WhatsApp
DAY-10.pdf
FREE AI Code Generator
Online Python Compiler
Gemini
programiz.com/python-programming/online-compiler/
Programiz Python Online Compiler
Programiz PRO
main.py
Share
Run
Output
Clear
def combination_sum(candidates, target):
"""
Finds all unique combinations of candidates that sum up to the target.
Args:
candidates: A list of distinct integers.
target: The target sum.
Returns:
A list of all unique combinations.
"""
result = []
candidates.sort()
def backtrack(combination, start, remain):
if remain == 0:
result.append(list(combination))
return
elif remain < 0:
return
for i in range(start, len(candidates)):
combination.append(candidates[i])
backtrack(combination, i, remain - candidates[i])
combination.pop()
backtrack([], 0, target)
return result
candidates = [2, 3, 6, 7]
target = 7
result = combination_sum(candidates, target)
print(result)
[[2, 2, 3], [7]]
=== Code Execution Successful ===
Search
ENG
IN
22:51
12-08-2024


(1) WhatsApp
DAY-10.pdf
FREE AI Code Generator
Online Python Compiler
Gemini
programiz.com/python-programming/online-compiler/
Programiz Python Online Compiler
Programiz PRO
main.py
Share
Run
Output
Clear
def combination_sum2(candidates, target):
"""
Finds all unique combinations of candidates that sum up to the target,
where each number in candidates can be used only once.
Args:
candidates: A list of distinct integers.
target: The target sum.
Returns:
A list of all unique combinations.
"""
result = []
candidates.sort()
def backtrack(combination, start, remain):
if remain == 0:
result.append(list(combination))
return
elif remain < 0:
return
for i in range(start, len(candidates)):
if i > start and candidates[i] == candidates[i - 1]:
continue # Skip duplicates
combination.append(candidates[i])
backtrack(combination, i + 1, remain - candidates[i])
combination.pop()
backtrack([], 0, target)
return result
candidates = [10, 1, 2, 7, 6, 1, 5]
target = 8
result = combination_sum2(candidates, target)
[[1, 1, 6], [1, 2, 5], [1, 7], [2, 6]]
=== Code Execution Successful ===
Search
ENG
IN
22:53
12-08-2024


(1) WhatsApp
FREE AI Code Generator: Gener
Online Python Compiler (Interp
Gemini
programiz.com/python-programming/online-compiler/
Programiz Python Online Compiler
Programiz PRO
main.py
Share
Run
Output
Clear
def permute(nums):
"""
Generates all possible permutations of a given array.
Args:
nums: The input array of distinct integers.
Returns:
A list of all possible permutations.
"""
def backtrack(first=0):
if first == n:
output.append(nums[:])
for i in range(first, n):
nums[first], nums[i] = nums[i], nums[first]
backtrack(first + 1)
nums[first], nums[i] = nums[i], nums[first]
n = len(nums)
output = []
backtrack()
return output
nums = [1, 2, 3]
result = permute(nums)
print(result)
[[1, 2, 3], [1, 3, 2], [2, 1, 3], [2, 3, 1], [3, 2, 1], [3, 1, 2]]
=== Code Execution Successful ===
29°
Search
ENG
IN
22:57
12-08-2024


(1) WhatsApp
FREE AI Code Generator
Online Python Compiler
Gemini
DAY-10.pdf
programiz.com/python-programming/online-compiler/
Programiz Python Online Compiler
Programiz PRO
main.py
Share
Run
Output
Clear
def permuteUnique(nums):
"""
Generates all unique permutations of a given array with duplicates.
Args:
nums: The input array of integers.
Returns:
A list of all unique permutations.
"""
def backtrack(first=0):
if first == n:
output.append(nums[:])
for i in range(first, n):
if i > first and nums[i] == nums[i - 1]:
continue
nums[first], nums[i] = nums[i], nums[first]
backtrack(first + 1)
nums[first], nums[i] = nums[i], nums[first]
n = len(nums)
output = []
nums.sort()
backtrack()
return output
[[1, 1, 2], [1, 2, 1], [2, 1, 1]]
=== Code Execution Successful ===
Search
ENG
IN
22:59
12-08-2024