

## N QUEEN

```
n queen.py - C:\Users\Dharani M\AppData\Local\Programs\Python\Python312\n queen.py (3.12.3)
File Edit Format Run Options Window Help
def solve_n_queens(n):
    def is_valid(board, row, col):
        for i in range(row):
            if board[i] == col or \
                board[i] - i == col - row or \
                board[i] + i == col + row:
                return False
            return True
    def place_queens(n, row, board):
        if row == n:
            result.append(board[:])
            return
        for col in range(n):
            if is_valid(board, row, col):
                board[row] = col
                place_queens(n, row + 1, board)
        result = []
        place_queens(n, 0, [-1]*n)
        return [{"*"*i + "Q" + "*"*(n-i-1) for i in sol for sol in result}]
n = 4
for solution in solve_n_queens(n):
    for row in solution:
        print(row)
    print()

Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\Dharani M\AppData\Local\Programs\Python\Python312\n queen.py
.Q..
...Q
Q...
..Q.
..Q.
Q...
.Q..
>>>
```

## Largest palindrome sequence

```
longest palindrome.py - C:\Users\Dharani M\AppData\Local\Programs\Python\Python312\longest palindrome...
File Edit Format Run Options Window Help
def longest_palindrome(s):
    n = len(s)
    if n == 0:
        return ""
    dp = [[0] * n for _ in range(n)]
    start = 0
    max_length = 1
    for i in range(n):
        dp[i][i] = 1
    for i in range(n - 1):
        if s[i] == s[i + 1]:
            dp[i][i + 1] = 1
            start = i
            max_length = 2
    for length in range(3, n + 1):
        for i in range(n - length + 1):
            j = i + length - 1
            if s[i] == s[j] and dp[i + 1][j - 1] == 1:
                dp[i][j] = dp[i + 1][j - 1] + 2
                if dp[i][j] > max_length:
                    max_length = dp[i][j]
                    start = i
    for i in range(n):
        for j in range(i + 1, n):
            dp[i][j] = max(dp[i][j], dp[i + 1][j], dp[i][j - 1])
    print("DP Table:")
    for row in dp:
        print(row)
    return s[start:start + max_length]
s = "teeth"
print("Input string:", s)
print("The longest palindrome substring is:", longest_palindrome(s))

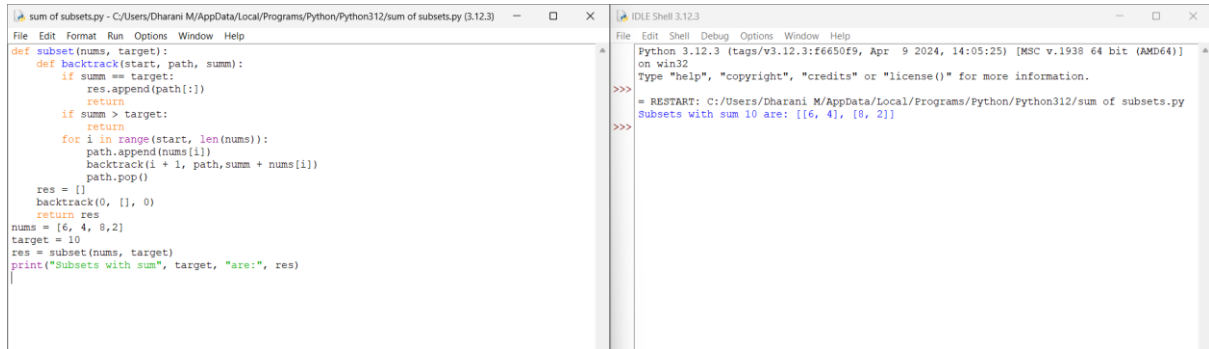
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\Dharani M\AppData\Local\Programs\Python\Python312\longest palindrome
.py
Input string: teeth
DP Table:
[[1, 1, 2, 4, 4],
 [0, 1, 2, 2, 2],
 [0, 0, 1, 1, 1],
 [0, 0, 0, 1, 1],
 [0, 0, 0, 0, 1]]
The longest palindrome substring is: teet
>>>
```

## Sum of subset

```
graph coloring.py - C:\Users\Dharani M\AppData\Local\Programs\Python\Python312\graph coloring.py (3.12.3)
File Edit Format Run Options Window Help
def is_safe(graph, colors, v, c):
    for i in range(len(graph)):
        if graph[v][i] == 1 and colors[i] == c:
            return False
    return True
def graph_coloring_util(graph, m, colors, v, num_vertices, all_colorings):
    if v == num_vertices:
        all_colorings.append(colors[:])
        return
    for c in range(1, m + 1):
        if is_safe(graph, colors, v, c):
            colors[v] = c
            graph_coloring_util(graph, m, colors, v + 1, num_vertices, all_colorings)
            colors[v] = 0
def graph_coloring(graph, m):
    num_vertices = len(graph)
    colors = [0] * num_vertices
    all_colorings = []
    graph_coloring_util(graph, m, colors, 0, num_vertices, all_colorings)
    unique_colorings = []
    for coloring in all_colorings:
        if coloring not in unique_colorings:
            unique_colorings.append(coloring)
    if unique_colorings[0] == unique_colorings[-1]:
        return unique_colorings[:-1]
    return unique_colorings
graph = [
    [0, 1, 1, 0],
    [1, 0, 0, 0],
    [1, 0, 0, 1],
    [0, 0, 1, 0]
]
m = 3
unique_colorings = graph_coloring(graph, m)
print("All unique colorings:")
for coloring in unique_colorings:
    print(coloring)

Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\Dharani M\AppData\Local\Programs\Python\Python312\graph coloring.py
All unique colorings:
[[1, 2, 1],
 [1, 2, 2, 3],
 [1, 2, 3, 1],
 [1, 2, 3, 2],
 [1, 3, 2, 1],
 [1, 3, 2, 3],
 [1, 3, 3, 1],
 [1, 3, 3, 2],
 [2, 1, 1, 2],
 [2, 1, 1, 3],
 [2, 1, 3, 1],
 [2, 1, 3, 2],
 [2, 3, 1, 2],
 [2, 3, 1, 3],
 [2, 3, 3, 1],
 [2, 3, 3, 2],
 [3, 1, 1, 2],
 [3, 1, 1, 3],
 [3, 1, 2, 1],
 [3, 1, 2, 3],
 [3, 2, 1, 2],
 [3, 2, 1, 3],
 [3, 2, 2, 1],
 [3, 2, 2, 3]]
>>>
```

## Graph coloring



The image shows a screenshot of a Python IDE with two windows. The left window, titled 'sum of subsets.py', contains a recursive function 'subset' that finds all subsets of a given list of numbers that sum up to a target value. The function uses a backtracking approach. The right window, titled 'IDLE Shell 3.12.3', shows the execution of the script, which prints the subsets of the list [6, 4, 8, 2] that sum up to 10.

```
sum of subsets.py - C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/sum of subsets.py (3.12.3)
File Edit Format Run Options Window Help
def subset(nums, target):
    def backtrack(start, path, summ):
        if summ == target:
            res.append(path[:])
            return
        if summ > target:
            return
        for i in range(start, len(nums)):
            path.append(nums[i])
            backtrack(i + 1, path, summ + nums[i])
            path.pop()
        res = []
        backtrack(0, [], 0)
    return res
nums = [6, 4, 8, 2]
target = 10
res = subset(nums, target)
print("Subsets with sum", target, "are:", res)

IDLE Shell 3.12.3
File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/sum of subsets.py
Subsets with sum 10 are: [[6, 4], [8, 2]]
>>>
```